```
DDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBB      UUU         UUU     GGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG   GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG   GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG   GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG         GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG         GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG         GGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB      UUUUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB      UUUUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB      UUUUUUUUUUUUUUUUU     GGGGGGGGG
```

```
RRRRRRRR      SSSSSSSS  TTTTTTTTTT    CCCCCCCC  NN      NN  TTTTTTTTTT  RRRRRRRR    LL
RRRRRRRR      SSSSSSSS  TTTTTTTTTT    CCCCCCCC  NN      NN  TTTTTTTTTT  RRRRRRRR    LL
RR      RR  SS              TT      CC          NN      NN      TT      RR      RR  LL
RR      RR  SS              TT      CC          NN      NN      TT      RR      RR  LL
RR      RR  SS              TT      CC          NNNN    NN      TT      RR      RR  LL
RR      RR  SS              TT      CC          NNNN    NN      TT      RR      RR  LL
RRRRRRRR      SSSSSS        TT      CC          NN  NN  NN      TT      RRRRRRRR    LL
RRRRRRRR      SSSSSS        TT      CC          NN  NN  NN      TT      RRRRRRRR    LL
RR  RR              SS      TT      CC          NN    NNNN      TT      RR  RR      LL
RR  RR              SS      TT      CC          NN    NNNN      TT      RR  RR      LL
RR    RR            SS      TT      CC          NN      NN      TT      RR    RR    LL              ....
RR    RR            SS      TT      CC          NN      NN      TT      RR    RR    LL              ....
RR      RR  SSSSSSSS        TT        CCCCCCCC  NN      NN      TT      RR      RR  LLLLLLLLLL      ....
RR      RR  SSSSSSSS        TT        CCCCCCCC  NN      NN      TT      RR      RR  LLLLLLLLLL      ....

LL              IIIIII      SSSSSSSS
LL              IIIIII    SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

```
;     1          0001  0 MODULE RSTCNTRL (IDENT = 'V04-000') =
;     2          0002  0
;     3          0003  1 BEGIN
;     4          0004  1
;     5          0005  1 !**********************************************************************
;     6          0006  1 !*                                                                    *
;     7          0007  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                           *
;     8          0008  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.            *
;     9          0009  1 !*  ALL RIGHTS RESERVED.                                             *
;    10          0010  1 !*                                                                    *
;    11          0011  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    12          0012  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    13          0013  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
;    14          0014  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
;    15          0015  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
;    16          0016  1 !*  TRANSFERRED.                                                       *
;    17          0017  1 !*                                                                    *
;    18          0018  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
;    19          0019  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
;    20          0020  1 !*  CORPORATION.                                                       *
;    21          0021  1 !*                                                                    *
;    22          0022  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    23          0023  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.            *
;    24          0024  1 !*                                                                    *
;    25          0025  1 !*                                                                    *
;    26          0026  1 !**********************************************************************
;    27          0027  1 !
;    28          0028  1 ! WRITTEN BY
;    29          0029  1 !       Bert Beander    June, 1980.
;    30          0030  1 !
;    31          0031  1 ! MODULE FUNCTION
;    32          0032  1 !       This module contains the routines which build and remove DEBUG's
;    33          0033  1 !       Run-Time Symbol Table (RST).  This includes the routines which
;    34          0034  1 !       initialize the RST when DEBUG is started up as well as the routines
;    35          0035  1 !       which build and remove (SET and CANCEL) a specific module's RST
;    36          0036  1 !       during the debugging session.  The routines which maintain the
;    37          0037  1 !       DEBUG Scope List are also here.
;    38          0038  1 !
;    39          0039  1 ! MODIFIED BY
;    40          0040  1 !       Rich Title      Nov, 1981       Fixed a bug in DBG$RST_SETSCOPE :
;    41          0041  1 !                                       SET SCOPE MOD1 sometimes failed if
;    42          0042  1 !                                       MOD1 is a module which is already
;    43          0043  1 !                                       set.
;    44          0044  1 !
;    45          0045  1 !       Ping Sager      Mar, 1982       Break original DBG$RST_SETSCOPE
;    46          0046  1 !                                       routine into two parts.  One builds
;    47          0047  1 !                                       the scope list, the other sets up
;    48          0048  1 !                                       the scope list.
;    49          0049  1 !
;    50          0050  1 !       Rich Title      Mar, 1982       Added to RST_BUILD_ERROR_HANDLER
;    51          0051  1 !                                       some code that marks the module
;    52          0052  1 !                                       as not being set, if an error occured
;    53          0053  1 !                                       when trying to set it.
;    54          0054  1 !
;    55          0055  1 !       Ping Sager      Apr, 1982       Added DBG$GST_INIT.
;    56          0056  1 !
;    57          0057  1 !       Ping Sager      Apr, 1982       Insert TYPE RST into hash chain also
```

```
;    58          0058  1 !
;    59          0059  1 !       V. Holt         Jul, 1982       Modified STATIC_CHECK to accept static
;    60          0060  1 !                                       records.  Modified DBG$RST_INIT so that
;    61          0061  1 !                                       static records are added to the program
;    62          0062  1 !                                       level SAT.
;    63          0063  1 !
;    64          0064  1 !       Bert Beander    Dec, 1982       Put in the fast initialization of the
;    65          0065  1 !                                       RST using the Debug Module Table (DMT).
;    66          0066  1 !
;    67          0067  1 !       Rich Title      Feb 1983        Added DBG$ANALYZE_HASH to aid in
;    68          0068  1 !                                       analysis of hashing algorithms.
;    69          0069  1 !
;    70          0070  1 !       Rich Title      Feb 1983        Added BUILD_TYPE_TABLE flag to suppress
;    71          0071  1 !                                       unnecessary looping through RST, and
;    72          0072  1 !                                       thus speed up SET MODULE in some cases.
;    73          0073  1 !
;    74          0074  1 !       Rich Title      Feb 1983        Removed the CHECK_DUPLICATE routine
;    75          0075  1 !                                       which screened out duplicate BLISS
;    76          0076  1 !                                       RST entries during the RST build.
;    77          0077  1 !                                       The was costing us very heavily
;    78          0078  1 !                                       during the SET MODULE command. Now,
;    79          0079  1 !                                       this check is done during lookup
;    80          0080  1 !                                       (see the CHECK_DUPLICATE routine
;    81          0081  1 !                                       in RSTACCESS).
;    82          0082  1 !       Rich Title      Apr 1983        Increased size of hash table in order
;    83          0083  1 !                                       to speed up symbol lookup.
;    84          0084  1 !       Rich Title      Apr 1983        Made changes to SET SCOPE to handle
;    85          0085  1 !                                       SET SCOPE/MODULE.
;    86          0086  1 !       Rich Title      Apr 1983        Added SET MODULE/ALLOCATE
;    87          0087  1 !
;    88          0088  1 !       Ping Sager      May 1983        Added shareable image support
;    89          0089  1 !
;    90          0090  1 !       Walter Carrell  Oct 1983        Added support for continuation DST records
;    91          0091  1 !
;    92          0092  1 !       Walter Carrell  Nov 1983        Added overloaded symbol support
;    93          0093  1 !
;    94          0094  1 !       Brad Becker     Nov 1983        Added code to hash in type spec. DSTs
;    95          0095  1 !
;    96          0096  1
;    97          0097  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    98          0231  1
;    99          0232  1 FORWARD ROUTINE
;   100          0233  1     DBG$ANALYZE_HASH:   NOVALUE,        ! Dump out information about hash chains
;   101          0234  1     DBG$GET_DST_NAME,                   ! Find the name string in a DST record
;   102          0235  1     DBG$GST_INIT,                       ! Build RST entries for all global
;   103          0236  1                                         !       symbols in GST
;   104          0237  1     DBG$GST_TO_RST:     NOVALUE,        ! Create RST entry for a global symbol
;   105          0238  1     DBG$HASH_FIND,                      ! Find a name in the RST Hash Table
;   106          0239  1     DBG$HASH_FIND_SETUP:NOVALUE,        ! Set up calls on DBG$HASH_FIND routine
;   107          0240  1     DBG$HASH_INSERT:    NOVALUE,        ! Insert an RST entry in the Hash Table
;   108          0241  1     DBG$RST_BUILD:      NOVALUE,        ! Build the RST for one module
;   109          0242  1     DBG$RST_BLDSCOPE_LIST,              ! Build the scope list
;   110          0243  1     DBG$RST_CANMOD,                     ! CANCEL a module in the RST
;   111          0244  1     DBG$RST_DST_LENGTH,                 ! Return the "real" DST length
;   112          0245  1     DBG$RST_DST_PTR,                    ! Return the "real" DST pointer
;   113          0246  1     DBG$RST_INIT:       NOVALUE,        ! Initialize DST and RST
;   114          0247  1     DBG$RST_MOST_RECENT:NOVALUE,        ! Mark the Most Recently Used module
```

```
;  115          0248  1      DBG$RST_NEXT_DST,                  ! Return the next DST given the one
;  116          0249  1      DBG$RST_REMOVE:     NOVALUE,       ! Remove a module from the RST
;  117          0250  1      DBG$RST_SETMOD,                    ! SET a module in the RST
;  118          0251  1      DBG$RST_SETSCOPE:   NOVALUE,       ! Set a list of scopes from SET SCOPE
;  119          0252  1      BLDSCOPE_ERROR_HANDLER,            ! Error handler for DBG$RST_BLDSCOPE_LIST
;  120          0253  1      BUILD_LABEL_RST,                   ! Build RST entries for numeric labels
;  121          0254  1      COPY_COMP_LIST:     NOVALUE,       ! Copy a record component list to RST
;  122          0255  1      COPY_VARIANT_LIST,                 ! Copy a record variant list to RST
;  123          0256  1      COMP_LIST_ADD_ENTRY:NOVALUE,       ! Add entry to a record component list
;  124          0257  1      GET_MEM,                           ! Get memory from the free memory pool
;  125          0258  1      GET_MOD_RST,                       ! Find a specified module in the RST
;  126          0259  1      GET_TYPEPTR:        NOVALUE,       ! Find record or enum type DST pointer
;  127          0260  1      MERGE_CONTINUATION_DST,            ! Move and merge continuation records
;  128          0261  1      POSITION_GST,                      ! Position GST pointer
;  129          0262  1      READ_DEBUG_MODULE_TABLE,           ! Read Debug Module Table to initialize
;  130          0263  1                                         !      the RST (for new images only)
;  131          0264  1      RST_BUILD_ERROR_HANDLER,           ! Error handler for DBG$RST_BUILD
;  132          0265  1      SHARED_IMAGE_NAMES: NOVALUE,       ! Make global symbols from all shared
;  133          0266  1                                         !      image names in this execution
;  134          0267  1      SHARED_IMAGE_NAMES_HANDLER,        ! Handler for above routine
;  135          0268  1      SHARED_IMAGE_NAMES_V3: NOVALUE,    ! Make global symbols from all shared
;  136          0269  1                                         !      image names in this execution
;  137          0270  1                                         !      for VMS 3.0 system
;  138          0271  1      STATIC_ADDRESS:     NOVALUE,       ! Make a Static Address Table entry
;  139          0272  1      STATIC_CHECK:       NOVALUE,       ! Check whether a data item is static
;  140          0273  1      STATIC_CHECK_HANDLER,              ! Handler for STATIC_CHECK
;  141          0274  1      TEST_DST_SIZE;                     ! Find DST size for Testable Debugger
```

```
;  143          0275  1 EXTERNAL ROUTINE
;  144          0276  1     DBG$EXPAND_MEMORY,                  ! Expand the free memory pool
;  145          0277  1     DBG$FIND_SEPTYP,                    ! Given SEPTYP DST, find the size
;  146          0278  1     DBG$FREE_MEM_LEFT,                  ! Return the free memory pool size
;  147          0279  1     DBG$GET_BITSIZE_FROM_TYPESPEC,      ! Get the bit size for this type spec
;  148          0280  1     DBG$GET_MEMORY,                     ! Get a memory block from memory pool
;  149          0281  1     DBG$GET_TEMPMEM,                    ! Get a memory block from the temp memory pool
;  150          0282  1     DBG$GET_BITSIZE_FROM_DESC,          ! Given Descr, find the size
;  151          0283  1     DBG$LANGUAGE,                       ! Return ASCIC name from a language code                    ;
;  152          0284  1     DBG$NCIS_ADD,
;  153          0285  1     DBG$NEWLINE:          NOVALUE,      ! DEBUG output routine
;  154          0286  1     DBG$NPATHDESC_TO_CS:NOVALUE,        ! Convert pathname to counted ASCII
;  155          0287  1     DBG$PRINT:            NOVALUE,      ! DEBUG output routine
;  156          0288  1     DBG$REL_MEMORY:       NOVALUE,      ! Release a block to free memory pool
;  157          0289  1     DBG$SCR_INITIALIZE: NOVALUE,        ! Initialize Screen Debugging
;  158          0290  1     DBG$SET_LANG: NOVALUE,              ! Set the current language setting
;  159          0291  1     DBG$SRC_INIT:         NOVALUE,      ! Initialize source line display blocks
;  160          0292  1     DBG$STA_GETSYMBOL:  NOVALUE,        ! Look up a symbol in the RST
;  161          0293  1     DBG$STA_LOCK_SYMID: NOVALUE,        ! Lock a SYMID in RST memory
;  162          0294  1     DBG$STA_LOOKUP_GBL,                 ! Look up a Global symbol in the GST
;  163          0295  1     DBG$STA_SYMNAME: NOVALUE,           ! Get the name string of a given symbol
;  164          0296  1     DBG$STA_UNLOCK_SYMID: NOVALUE,      ! Unlock a SYMID locked in RST memory
;  165          0297  1     DBG$TRANS_TYPE_CODE,                ! Returns the FCODE for the type spec
;  166          0298  1     SYS$CRMPSC,                         ! Create Map Section system service
;  167          0299  1     SYS$DELTVA;                         ! Delete Virtual memory space
;  168          0300  1
;  169          0301  1 EXTERNAL
;  170          0302  1     DBG$GB_LANGUAGE: BYTE,              ! The currently SET language code
;  171          0303  1     DBG$GL_MOVED_DST_LIST_HEAD,         ! The head of the Moved DST list
;  172          0304  1     DBG$GL_LIB_SIGNAL_ADDR,             ! Address of LIB$SIGNAL
;  173          0305  1     DBG$GL_LIB_STOP_ADDR,               ! Address of LIB$STOP
;  174          0306  1     DBG$GV_CONTROL : DBG$CONTROL_FLAGS, ! DEBUG control bits
;  175          0307  1     DBG$RUNFRAME : BLOCK[,BYTE],        ! The current user run frame
;  176          0308  1     DBG$GL_TRANSFER_ADDRESS,            ! Transfer address
;  177          0309  1
;  178          0310  1     ! Note - for FT1 we get the shareable image list from IAC$GL_IMAGE_LIST
;  179          0311  1     ! which is obtained by linking against SYS.STB. For FT2,
;  180          0312  1     ! we get it from SYS$GL_IMGLSTPTR which comes from STARLET.
;  181          0313  1     !
;  182          0314  1     SYS$GL_IMGLSTPTR,                   ! See SHARED_IMAGE_NAMES
;  183          0315  1     IAC$GL_IMAGE_LIST: WEAK,            ! See SHARED_IMAGE_NAMES
;  184          0316  1
;  185          0317  1     SYS$GL_IAFLINK: WEAK REF VECTOR[1]; ! Pointer to pointer to start of Image
;  186          0318  1                                         !      Activator Fixup Block List
;  187          0319  1
;  188          0320  1 GLOBAL
;  189          0321  1     DST$BEGIN_ADDR: INITIAL(0),         ! Virtual address where the DST begins
;  190          0322  1     DST$END_ADDR: INITIAL(0),           ! Virtual address of last byte of DST
;  191          0323  1     DST$NEXT_ADDR,                      ! -
;  192          0324  1     GST$BEGIN_ADDR: INITIAL(0),         ! Virtual address where the GST (Global
;  193          0325  1                                         !      Symbol Table) begins
;  194          0326  1     GST$NEXT_ADDR,                      ! -
;  195          0327  1     GST$REC_COUNT: INITIAL(0),          ! Number of records in the GST
;  196          0328  1     LRUM$LISTHEAD: REF LRUMSENTRY       ! Pointer to the list head for the Least
;  197          0329  1                          INITIAL(0),    !      Recently Used Module (LRUM) chain
;  198          0330  1     LRUM$MOST_RECENT: INITIAL(0),       ! Pointer to the Module RST Entry of the
;  199          0331  1                                         !      Most Recently Used module
```

```
;  200          0332  1     RST$REF_LIST: REF VECTOR[,LONG]          ! Pointer to a list of pointers to RST
;  201          0333  1                       INITIAL(0),            !      entries referenced by the current
;  202          0334  1                                              !      command.
;  203          0335  1     RST$SET_SCOPE: INITIAL(FALSE),           ! Flag set to TRUE if DBG$STA_GETSYMBOL
;  204          0336  1                                              !      is called from DBG$RST_SETSCOPE
;  205          0337  1     RST$START_ADDR: INITIAL(0),              ! Address of the first Module RST Entry
;  206          0338  1                                              !      on the RST module chain.
;  207          0339  1     RST$TEMP_LIST: INITIAL(0),               ! Pointer to Temporary RST Entry List
;  208          0340  1     SAT$START_ADDR: INITIAL(0),              ! Address of first Static Address Table
;  209          0341  1                                              !      (SAT) entry on Program SAT chain
;  210          0342  1     SCOPE$LIST: REF SCOPE$ENTRY;             ! Pointer to current Scope List
;  211          0343  1
;  212          0344  1 LITERAL
;  213          0345  1     GST_TYPE              = 1,               ! Record type is GST
;  214          0346  1     GST_RECORD_TYPE       = 0,               ! Type of GST record
;  215          0347  1     MIN_MODS_IN_RST       = 6,               ! Minimum number of modules the RST must
;  216          0348  1                                              !      be able to hold--in particular,
;  217          0349  1                                              !      the MIN_MODS_IN_RST largest mod-
;  218          0350  1                                              !      ules must fit in the RST.
;  219          0351  1     RST$K_HASHTBLSIZE     = 2503;            ! The size of the RST Hash Table
;  220          0352  1                                              !      (A prime number should be used).
;  221          0353  1
;  222          0354  1 OWN
;  223          0355  1     DBG$HASH_INDEX,                          ! Hash index used to communicate between
;  224          0356  1                                              !      routines DBG$HASH_FIND_SETUP and
;  225          0357  1                                              !      DBG$HASH_FIND
;  226          0358  1     DBG$HASH_RSTPTR: REF RST$ENTRY           ! Pointer to current RST entry on the
;  227          0359  1                   INITIAL(0),                !      current hash chain--used to com-
;  228          0360  1                                              !      municate between DBG$HASH_FIND
;  229          0361  1                                              !      and DBG$HASH_FIND_SETUP
;  230          0362  1     EXEFAB: $FAB(),                          ! RMS File Access Block (FAB) for the
;  231          0363  1                                              !      executable image file
;  232          0364  1     EXERAB: $RAB(),                          ! RMS Record Access Block (RAB) for the
;  233          0365  1                                              !      executable image file
;  234          0366  1     IMAGE_MODULE_RST_SIZE,                   ! Keep track the module RST size of the image
;  235          0367  1     IMAGE_MODULE_END_ADDR,                   ! Keep track the end address of the module
;  236          0368  1     MOD_BEING_BUILT,                         ! Pointer to RST entry for module being
;  237          0369  1                                              !      built--used to communicate with
;  238          0370  1                                              !      error handler.
;  239          0371  1     PREV_RST_PTR: REF RST$ENTRY,             ! Pointer to the previous RST entry in
;  240          0372  1                                              !      the chain of Module RST entries
;  241          0373  1     RST$HASH_TABLE:                          ! The RST Hash Table--a vector of list
;  242          0374  1         VECTOR[2*RST$K_HASHTBLSIZE],         !      heads for RST hash chains.
;  243          0375  1     TYPRSTLOC: REF VECTOR[,LONG],            ! Pointer to next available location in
;  244          0376  1                                              !      the current Type RST Entry block
;  245          0377  1     TYPRSTPTR: REF RST$ENTRY;                ! Pointer to the current Type RST Entry
```

```
;  247          0378  1 GLOBAL ROUTINE DBG$ANALYZE_HASH: NOVALUE =
;  248          0379  1 !
;  249          0380  1 ! FUNCTION
;  250          0381  1 !       Dumps information about the length of the hash chains. This
;  251          0382  1 !       information is useful in analyzing how good our hash algorithm is.
;  252          0383  1 !       This routine should probably be commented out in the version of
;  253          0384  1 !       the debugger that we ship since it is only used to aid our
;  254          0385  1 !       own development.
;  255          0386  1 !
;  256          0387  1 ! INPUTS
;  257          0388  1 !       The entire RST is scanned.
;  258          0389  1 !
;  259          0390  1 ! OUTPUTS
;  260          0391  1 !       The length of each hash chain is displayed at the terminal.
;  261          0392  1 !
;  262          0393  1     INCR I FROM 0 TO RST$K_HASHTBLSIZE-1 DO
;  263          0394  2         BEGIN
;  264          0395  2         LOCAL
;  265          0396  2             COUNT,
;  266          0397  2             NAME,
;  267          0398  2             NEW_RSTPTR: REF RST$ENTRY,
;  268          0399  2             RSTPTR: REF RST$ENTRY;
;  269          0400  2         DBG$PRINT(UPLIT BYTE(%ASCIC 'Hash chain !UL:'), .I );
;  270          0401  2         DBG$NEWLINE();
;  271          0402  2         RSTPTR = .RST$HASH_TABLE[2*.I];
;  272          0403  2         NEW_RSTPTR = .RSTPTR[RST$L_HASH_FLINK];
;  273          0404  2         COUNT = 0;
;  274          0405  2         WHILE .NEW_RSTPTR NEQ .RSTPTR DO
;  275          0406  3             BEGIN
;  276          0407  3             COUNT = .COUNT + 1;
;  277          0408  3             DBG$STA_SYMNAME(.NEW_RSTPTR[RST$L_HASH_BLINK], NAME );
;  278          0409  3             DBG$PRINT( .NAME );
;  279          0410  3             DBG$NEWLINE();
;  280          0411  3             NEW_RSTPTR = .NEW_RSTPTR[RST$L_HASH_FLINK];
;  281          0412  2             END;
;  282          0413  2         DBG$PRINT(UPLIT BYTE(%ASCIC '    !UL entries.'), .COUNT);
;  283          0414  2         DBG$NEWLINE();
;  284          0415  1         END;


                                                                .TITLE  RSTCNTRL
                                                                .IDENT  \V04-000\

                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

4C 55 21 20 6E 69 61 68 63 20 68 73 61 48 0F  00000 P.AAA:      .ASCII  <15>\Hash chain !UL:\                   ;
                                          3A  0000F
   2E 73 65 69 72 74 6E 65 20 4C 55 21 09 0D  00010 P.AAB:      .ASCII  <13><9>\!UL entries.\                   ;

                                                                .PSECT  DBG$OWN,NOEXE,  PIC,2

                                              00000 DBG$HASH_INDEX:
                                                                .BLKB   4
                                  00000000    00004 DBG$HASH_RSTPTR:
                                                                .LONG   0
                                        03    00008 EXEFAB:     .BYTE   3                                       ;
                                        50    00009             .BYTE   80                                      ;
```

```
                0000  0000A           .WORD   0
            00000000  0000C           .LONG   0
            00000000  00010           .LONG   0
            00000000  00014           .LONG   0
            00000000  00018           .LONG   0
                0000  0001C           .WORD   0
                  02  0001E           .BYTE   2
                  00  0001F           .BYTE   0
            00000000  00020           .LONG   0
                  00  00024           .BYTE   0
                  00  00025           .BYTE   0
                  00  00026           .BYTE   0
                  02  00027           .BYTE   2
            00000000  00028           .LONG   0
            00000000  0002C           .LONG   0
            00000000  00030           .LONG   0
            00000000  00034           .LONG   0
            00000000  00038           .LONG   0
                  00  0003C           .BYTE   0
                  00  0003D           .BYTE   0
                0000  0003E           .WORD   0
            00000000  00040           .LONG   0
                0000  00044           .WORD   0
                  00  00046           .BYTE   0
                  00  00047           .BYTE   0
            00000000  00048           .LONG   0
            00000000  0004C           .LONG   0
                0000  00050           .WORD   0
                  00  00052           .BYTE   0
                  00  00053           .BYTE   0
            00000000  00054           .LONG   0
                  01  00058 EXERAB:    .BYTE   1
                  44  00059           .BYTE   68
                0000  0005A           .WORD   0
            00000000  0005C           .LONG   0
            00000000  00060           .LONG   0
            00000000  00064           .LONG   0
                0000# 00068           .WORD   0[3]
                0000  0006E           .WORD   0
            00000000  00070           .LONG   0
                0000  00074           .WORD   0
                  00  00076           .BYTE   0
                  00  00077           .BYTE   0
                0000  00078           .WORD   0
                0000  0007A           .WORD   0
            00000000  0007C           .LONG   0
            00000000  00080           .LONG   0
            00000000  00084           .LONG   C
            00000000  00088           .LONG   0
                  00  0008C           .BYTE   0
                  00  0008D           .BYTE   0
                  00  0008E           .BYTE   0
                  00  0008F           .BYTE   0
            00000000  00090           .LONG   0
            00000000  00094           .LONG   0
            00000000  00098           .LONG   0
                      0009C IMAGE_MODULE_RST_SIZE:
```

```
                        .BLKB   4
000A0 IMAGE_MODULE_END_ADDR:
                        .BLKB   4
000A4 MOD_BEING_BUILT:
                        .BLKB   4
000A8 PREV_RST_PTR:
                        .BLKB   4
000AC RST$HASH_TABLE:
                        .BLKB   20024
04EE4 TYPRSTLOC:
                        .BLKB   4
04EE8 TYPRSTPTR:
                        .BLKB   4

                        .PSECT  DBG$GLOBAL,NOEXE,  PIC,2

00000000  00000 DST$BEGIN_ADDR::
                        .LONG   0
00000000  00004 DST$END_ADDR::
                        .LONG   0
          00008 DST$NEXT_ADDR::
                        .BLKB   4
00000000  0000C GST$BEGIN_ADDR::
                        .LONG   0
          00010 GST$NEXT_ADDR::
                        .BLKB   4
00000000  00014 GST$REC_COUNT::
                        .LONG   0
00000000  00018 LRUM$LISTHEAD::
                        .LONG   0
00000000  0001C LRUM$MOST_RECENT::
                        .LONG   0
00000000  00020 RST$REF_LIST::
                        .LONG   0
00000000  00024 RST$SET_SCOPE::
                        .LONG   0
00000000  00028 RST$START_ADDR::
                        .LONG   0
00000000  0002C RST$TEMP_LIST::
                        .LONG   0
00000000  00030 SAT$START_ADDR::
                        .LONG   0
          00034 SCOPE$LIST::
                        .BLKB   4

                        .EXTRN  DBG$EXPAND_MEMORY
                        .EXTRN  DBG$FIND_SEPTYP
                        .EXTRN  DBG$FREE_MEM_LEFT
                        .EXTRN  DBG$GET_BITSIZE_FROM_TYPESPEC
                        .EXTRN  DBG$GET_MEMORY, DBG$GET_TEMPMEM
                        .EXTRN  DBG$GET_BITSIZE_FROM_DESC
                        .EXTRN  DBG$LANGUAGE, DBG$NCIS_ADD
                        .EXTRN  DBG$NEWLINE, DBG$NPATHDESC_TO_CS
                        .EXTRN  DBG$PRINT, DBG$REL_MEMORY
                        .EXTRN  DBG$SCR_INITIALIZE
                        .EXTRN  DBG$SET_LANG, DBG$SRC_INIT
                        .EXTRN  DBG$STA_GETSYMBOL
```

```
                        .EXTRN  DBG$STA_LOCK_SYMID
                        .EXTRN  DBG$STA_LOOKUP_GBL
                        .EXTRN  DBG$STA_SYMNAME
                        .EXTRN  DBG$STA_UNLOCK_SYMID
                        .EXTRN  DBG$TRANS_TYPE_CODE
                        .EXTRN  SYS$CRMPSC, SYS$DELTVA
                        .EXTRN  DBG$GB_LANGUAGE
                        .EXTRN  DBG$GL_MOVED_DST_LIST_HEAD
                        .EXTRN  DBG$GL_LIB_SIGNAL_ADDR
                        .EXTRN  DBG$GL_LIB_STOP_ADDR
                        .EXTRN  DBG$GV_CONTROL, DBG$RUNFRAME
                        .EXTRN  DBG$GL_TRANSFER_ADDRESS
                        .EXTRN  SYS$GL_IMGLSTPTR
                        .WEAK   LAC$GL_IMAGE_LIST
                        .WEAK   SYS$GL_IAFLINK

                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                  00FC 00000    .ENTRY  DBG$ANALYZE_HASH, Save R2,R3,R4,R5,R6,R7    ; 0378
57 00000000G   00   9E 00002    MOVAB   DBG$NEWLINE, R7
56 00000000G   00   9E 00009    MOVAB   DBG$PRINT, R6
5E             04   C2 00010    SUBL2   #4, SP
               52   D4 00013    CLRL    I                                          ; 0393
               52   DD 00015 1$: PUSHL   I                                         ; 0400
   00000000'   EF   9F 00017    PUSHAB  P.AAA
66             02   FB 0001D    CALLS   #2, DBG$PRINT
67             00   FB 00020    CALLS   #0, DBG$NEWLINE                            ; 0401
50         52  01   78 00023    ASHL    #1, I, R0                                  ; 0402
54 00000000'EF40   D0 00027    MOVL    RST$HASH_TABLE[R0], RSTPTR
53             64   D0 0002F    MOVL    (RSTPTR), NEW_RSTPTR                       ; 0403
               55   D4 00032    CLRL    COUNT                                      ; 0404
54             53   D1 00034 2$: CMPL   NEW_RSTPTR, RSTPTR                         ; 0405
               1B   13 00037    BEQL    3$
               55   D6 00039    INCL    COUNT                                      ; 0407
               5E   DD 0003B    PUSHL   SP                                         ; 0408
                 04 A3 DD 0003D    PUSHL   4(NEW_RSTPTR)
      00000000G  00   02 FB 00040    CALLS   #2, DBG$STA_SYMNAME
               6E   DD 00047    PUSHL   NAME                                       ; 0409
             66     01 FB 00049    CALLS   #1, DBG$PRINT
             67     00 FB 0004C    CALLS   #0, DBG$NEWLINE                        ; 0410
             53     63 D0 0004F    MOVL    (NEW_RSTPTR), NEW_RSTPTR               ; 0411
               E0   11 00052    BRB     2$                                         ; 0405
               55   DD 00054 3$: PUSHL   COUNT                                    ; 0413
   00000000'   EF   9F 00056    PUSHAB  P.AAB
             66     02 FB 0005C    CALLS   #2, DBG$PRINT
             67     00 FB 0005F    CALLS   #0, DBG$NEWLINE                        ; 0414
         AB  52 000009C6   8F   F3 00062    AOBLEQ  #2502, I, 1$                   ; 0393
                           04 0006A    RET                                        ; 0415

; Routine Size:  107 bytes,    Routine Base:  DBG$CODE + 0000
```

```
;  286          0416  1 GLOBAL ROUTINE DBG$GET_DST_NAME(DSTPTR) =
;  287          0417  1 !
;  288          0418  1 ! FUNCTION
;  289          0419  1 !       This routine accepts a pointer to a DST entry and returns the address of
;  290          0420  1 !       the counted string that contains the name of that DST symbol.  If there
;  291          0421  1 !       is no name field in the specified DST entry, an error is signalled.
;  292          0422  1 !
;  293          0423  1 ! INPUTS
;  294          0424  1 !       DSTPTR  - A pointer to the Debug Symbol Table entry whose name field is
;  295          0425  1 !                 to be located.
;  296          0426  1 !
;  297          0427  1 ! OUTPUTS
;  298          0428  1 !       The address of the counted string containing the symbol name is returned
;  299          0429  1 !                 as DBG$GET_DST_NAME's value.
;  300          0430  1 !
;  301          0431  1
;  302          0432  2     BEGIN
;  303          0433  2
;  304          0434  2     MAP
;  305          0435  2         DSTPTR: REF DST$RECORD;          ! Pointer to the input DST record
;  306          0436  2
;  307          0437  2     LOCAL
;  308          0438  2         BPTR: REF DST$BLI_TRAILER1;      ! Pointer to Bliss special cases DST
;  309          0439  2                                          !      record's first trailer
;  310          0440  2
;  311          0441  2
;  312          0442  2
;  313          0443  2     ! Use the DST record type as a CASE index.  For each type of DST record,
;  314          0444  2     ! we then locate the corresponding name field.
;  315          0445  2     !
;  316          0446  2     CASE .DSTPTR[DST$B_TYPE] FROM 0 TO 255 OF
;  317          0447  2         SET
;  318          0448  2
;  319          0449  2
;  320          0450  2         ! Handle all the DST record types of the "standard" format.
;  321          0451  2         !
;  322          0452  2         [DSC$K_DTYPE_LOWEST TO DSC$K_DTYPE_HIGHEST,
;  323          0453  2          DST$K_SEPTYP, DST$K_BOOL, DST$K_RECBEG,
;  324          0454  2          DST$K_VARBEG, DST$K_ENUMELT, DST$K_ENTRY,
;  325          0455  2          DST$K_LABEL, DST$K_[BLORLIT, DST$K_BLIFLD,
;  326          0456  2          DST$K_COB_HACK]:
;  327          0457  2             RETURN DSTPTR[DST$B_NAME];
;  328          0458  2
;  329          0459  2
;  330          0460  2         ! Handle the Module-Begin record.
;  331          0461  2         !
;  332          0462  2         [DST$K_MODBEG]:
;  333          0463  2             RETURN DSTPTR[DST$B_MODBEG_NAME];
;  334          0464  2
;  335          0465  2
;  336          0466  2         ! Handle the Routine-Begin record.
;  337          0467  2         !
;  338          0468  2         [DST$K_RTNBEG]:
;  339          0469  2             RETURN DSTPTR[DST$B_RTNBEG_NAME];
;  340          0470  2
;  341          0471  2
;  342          0472  2         ! Handle the Lexical Block-Begin record.
```

```
;  343          0473  2          !
;  344          0474  2          [DST$K_BLKBEG]:
;  345          0475  2              RETURN DSTPTR[DST$B_BLKBEG_NAME];
;  346          0476  2
;  347          0477  2
;  348          0478  2          ! Handle the PSECT record.
;  349          0479  2          !
;  350          0480  2          [DST$K_PSECT]:
;  351          0481  2              RETURN DSTPTR[DST$B_PSECT_NAME];
;  352          0482  2
;  353          0483  2
;  354          0484  2          ! Handle the Bliss Special Cases record.
;  355          0485  2          !
;  356          0486  2          [DST$K_BLI]:
;  357          0487  3              BEGIN
;  358          0488  3              BPTR = DSTPTR[DST$A_BLI_TRLR1] + .DSTPTR[DST$B_BLI_LNG];
;  359          0489  3              RETURN BPTR[DST$B_BLI_NAME];
;  360          0490  2              END;
;  361          0491  2
;  362          0492  2
;  363          0493  2          ! Handle the Enumeration Type Begin record.
;  364          0494  2          !
;  365          0495  2          [DST$K_ENUMBEG]:
;  366          0496  2              RETURN DSTPTR[DST$B_ENUMBEG_NAME];
;  367          0497  2
;  368          0498  2
;  369          0499  2          ! Handle the Type Specification DST record.
;  370          0500  2          !
;  371          0501  2          [DST$K_TYPSPEC]:
;  372          0502  2              RETURN DSTPTR[DST$B_TYPSPEC_NAME];
;  373          0503  2
;  374          0504  2
;  375          0505  2              ! Handle the Overloaded Symbol DST record.
;  376          0506  2              !
;  377          0507  2              [DST$K_OVERLOAD]:
;  378          0508  2                  RETURN DSTPTR[DST$B_OL_NAME];
;  379          0509  2
;  380          0510  2
;  381          0511  2              ! Handle all DST records which do not have symbol name fields by signal-
;  382          0512  2              ! ling an internal coding error.
;  383          0513  2              [INRANGE]:
;  384          0514  2                  $DBG_ERROR('RSTCNTRL\GET_DST_NAME 10');
;  385          0515  2
;  386          0516  2              TES;
;  387          0517  2
;  388          0518  2
;  389          0519  2          ! We should never get here--if we do, signal an internal coding error.
;  390          0520  2          !
;  391          0521  2          $DBG_ERROR('RSTCNTRL\GET_DST_NAME 20');
;  392          0522  2          RETURN 0;
;  393          0523  2
;  394          0524  1          END;


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0
```

```
44 5F 54 45 47 5C 4C 52 54 4E 43 54 53 52 18 0001E P.AAC:  .ASCII  <24>\RSTCNTRL\<92>\GET_DST_NAME 10\
                  30 31 20 45 4D 41 4E 5F 54 53 0002D
44 5F 54 45 47 5C 4C 52 54 4E 43 54 53 52 18 00037 P.AAD:  .ASCII  <24>\RSTCNTRL\<92>\GET_DST_NAME 20\
                  30 32 20 45 4D 41 4E 5F 54 53 00046

                                                           .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                0004 00000 .ENTRY  DBG$GET_DST_NAME, Save R2                ; 0416
                                      52 00000000G   00  9E 00002 MOVAB  LIB$SIGNAL, R2
                                      50       04    AC  D0 00009 MOVL   DSTPTR, R0                       ; 0446
                                FF    8F       00    01  A0  8F 0000D CASEB  1(R0), #0, #255
          0200        0200        0200        0206        00013 1$:  .WORD  3$-1$,-
          0200        0200        0200        0200        0001B      2$-1$,-
          0200        0200        0200        0200        00023      2$-1$,-
          0200        0200        0200        0200        0002B      2$-1$,-
          0200        0200        0200        0200        00033      2$-1$,-
          0200        0200        0200        0200        0003B      2$-1$,-
          0200        0200        0200        0200        00043      2$-1$,-
          0200        0200        0200        0200        0004B      2$-1$,-
          0200        0200        0200        0200        00053      2$-1$,-
          0225        0225        0200        0200        0005B      2$-1$,-
          0225        0225        0225        0225        00063      2$-1$,-
          0225        0225        0225        0225        0006B      2$-1$,-
          0225        0225        0225        0225        00073      2$-1$,-
          0225        0225        0225        0225        0007B      2$-1$,-
          0225        0225        0225        0225        00083      2$-1$,-
          0225        0225        0225        0225        0008B      2$-1$,-
          0225        0225        0225        0225        00093      2$-1$,-
          0225        0225        0225        0225        0009B      2$-1$,-
          0225        0225        0225        0225        000A3      2$-1$,-
          0225        0225        0225        0225        000AB      2$-1$,-
          0225        0225        0225        0225        000B3      2$-1$,-
          0225        0225        0225        0225        000BB      2$-1$,-
          0225        0225        0225        0225        000C3      2$-1$,-
          0225        0225        0225        0225        000CB      2$-1$,-
          0225        0225        0225        0225        000D3      2$-1$,-
          0225        0225        0225        0225        000DB      2$-1$,-
          0225        0225        0225        0225        000E3      2$-1$,-
          0225        0225        0225        0225        000EB      2$-1$,-
          0225        0225        0225        0225        000F3      2$-1$,-
          0225        0225        0225        0225        000FB      2$-1$,-
          0225        0225        0225        0225        00103      2$-1$,-
          0225        0225        0225        0225        0010B      2$-1$,-
          0225        0225        0225        0225        00113      2$-1$,-
          0225        0225        0225        0225        0011B      2$-1$,-
          0225        0225        0225        0225        00123      2$-1$,-
          0225        0225        0225        0225        0012B      2$-1$,-
          0225        0225        0225        0225        00133      2$-1$,-
          0225        0225        0225        0225        0013B      2$-1$,-
          0225        0225        0225        0225        00143      8$-1$,-
          0225        0200        0225        0225        0014B      8$-1$,-
          0200        0225        0225        0225        00153      8$-1$,-
          0200        0225        0213        0200        0015B      8$-1$,-
          0200        0225        0221        0225        00163      8$-1$,-
          0219        0225        0225        0225        0016B      8$-1$,-
```

```
0225    0200    0225    0200    00173
0200    0225    0200    0225    0017B
0200    0200    0225    0200    00183
0225    0200    0225    0200    0018B
0225    0225    0225    0225    00193
0225    0225    0225    0225    0019B
0225    0225    0225    0225    001A3
0225    0225    0225    0225    001AB
0225    0225    0225    0225    001B3
0225    0225    0225    0225    001BB
0225    0225    0225    0225    001C3
0225    0225    0225    0225    001CB
0225    0225    0225    0225    001D3
0225    0225    0225    0225    001DB
0225    0225    0225    0225    001E3
0225    0225    0225    0225    001EB
0225    0225    0225    0225    001F3
0225    0225    0225    0225    001FB
0225    0225    0225    0225    00203
0225    0225    0225    0225    0020B
```

```
8$-1$,-
8$-1$,-
8$-1$,-
```

```
8$-1$,-
8$-1$,-
8$-1$,-
```

```
2$-1$,-
8$-1$,-
8$-1$,-
8$-1$,-
8$-1$,-
2$-1$,-
2$-1$,-
4$-1$,-
8$-1$,-
2$-1$,-
8$-1$,-
7$-1$,-
8$-1$,-
2$-1$,-
8$-1$,-
8$-1$,-
8$-1$,-
5$-1$,-
2$-1$,-
8$-1$,-
2$-1$,-
8$-1$,-
8$-1$,-
2$-1$,-
8$-1$,-
2$-1$,-
2$-1$,-
8$-1$,-
2$-1$,-
8$-1$,-
2$-1$,-
2$-1$,-
2$-1$,-
8$-1$,-
2$-1$,-
8$-1$,-
8$-1$,-
8$-1$,-
8
```

```
                                                                                8$-1$,-
                                                                                8$-1$,-
                                                                                8$-1$,-
```

```
                62               03  FB 00246          CALLS   #3, LIB$SIGNAL                          ;
                   00000000'     EF  9F 00249          PUSHAB  P.AAD                                   ; 0521
                                 01  DD 0024F          PUSHL   #1                                      ;
                   00028362      8F  DD 00251          PUSHL   #164706                                 ;
                62               03  FB 00257          CALLS   #3, LIB$SIGNAL                          ;
                                 50  D4 0025A          CLRL    R0                                      ; 0522
                                     04 0025C          RET                                             ; 0524

; Routine Size:  605 bytes,    Routine Base:  DBG$CODE + 006B
```

```
;  396     0525  1 GLOBAL ROUTINE DBG$GST_INIT(IMGFILCHAN, BASEADDR, MODRSTPTR): =
;  397     0526  1 !
;  398     0527  1 ! FUNCTION
;  399     0528  1 !       This routine sequentially reads in the Global Symbols and calls
;  400     0529  1 !       DBG$GST_TO_RST to build RST entry for each Global Symbol at RST
;  401     0530  1 !       intialization time in DBG$RST_INIT.
;  402     0531  1 !
;  403     0532  1 !       Before this routine is called, the location of the GST in the image
;  404     0533  1 !       file is found, and it is mapped into DEBUG's image.  The begin addrss
;  405     0534  1 !       of this buffer is pointed by GST$BEGIN_ADDR.  This routine analyzes
;  406     0535  1 !       the GST record, and moves through the buffer, getting the buffer
;  407     0536  1 !       address of each global symbol entry as it is seen.  When the buffer
;  408     0537  1 !       is exhausted, this routine reads in the next GST record.  It halts at
;  409     0538  1 !       end of file and returns a value of zero to the caller.
;  410     0539  1 !
;  411     0540  1 !       The format of one of these concatenated records is a single
;  412     0541  1 !       leading type containing the value 1, indicating that the record is
;  413     0542  1 !       indeed a GST record.  Each entry in the record has a fixed number of
;  414     0543  1 !       overhead types followed by a symbol name that is a variable number
;  415     0544  1 !       of bytes.  The entries we are interested in processing are the
;  416     0545  1 !       global symbol definitions and entry point symbol and prodecure definitions.
;  417     0546  1 !       The other defined type, PSECT definition, is noted only because it must
;  418     0547  1 !       be successfully passed over.  The detailed format of the GST is defined
;  419     0548  1 !       in DBGLIB.
;  420     0549  1 !
;  421     0550  1 ! INPUT
;  422     0551  1 !       IMGFILCHAN      - The channel number that the image is opened on.
;  423     0552  1 !
;  424     0553  1 !       BASEADDR        - Base address of the shareable image. (it may be zero).
;  425     0554  1 !
;  426     0555  1 !       MODRSTPTR       - Module RST pointer which contains all global symbols.
;  427     0556  1 !
;  428     0557  1 ! OUTPUT
;  429     0558  1 !       RST entries for all Global Symbols are built.
;  430     0559  1 !
;  431     0560  1
;  432     0561  2     BEGIN
;  433     0562  2
;  434     0563  2     MAP
;  435     0564  2         MODRSTPTR: REF RST$ENTRY;       ! Pointer to anonymous module RST entry
;  436     0565  2                                         !       or shareable image module RST
;  437     0566  2
;  438     0567  2     LOCAL
;  439     0568  2         EXE_FILNAM_DESC: VECTOR[2,LONG],! File name descriptor for $CRMPSC
;  440     0569  2         EXE_HDRBOUNDS: VECTOR[2,LONG],  ! Map section bounds descriptor for the
;  441     0570  2                                         !       $CRMPSC system service
;  442     0571  2         EXE_SECBOUNDS: VECTOR[2,LONG],  ! Map section bounds descriptor for the
;  443     0572  2                                         !       $CRMPSC system service
;  444     0573  2         EXE_SECNAM_DESC: VECTOR[2,LONG],! Map section name descriptor for the
;  445     0574  2                                         !       $CRMPSC system service
;  446     0575  2         GSTADDR: REF VECTOR[,WORD],     ! Pointer to Gst record-used to find
;  447     0576  2                                         !       first GST data record
;  448     0577  2         GSTPTR: REF BLOCK[,BYTE],       ! Virtual address of current GST record
;  449     0578  2         IMG_DST_BLOCK: REF IHS$ENTRY,   ! Pointer to DST/GST descriptor in the
;  450     0579  2                                         !       executable image header
;  451     0580  2         IMG_HEADER: REF BLOCK[,BYTE],   ! Pointer to the .EXE file image header
;  452     0581  2         NUM_ARGS: BYTE,                 ! maximum number of arguments.
```

```
;  453          0582  2          SATPTR: REF SAT$ENTRY,          ! Pointer to Program Statci address table
;  454          0583  2          STATUS: BLOCK[1,LONG];
;  455          0584  2
;  456          0585  2
;  457          0586  2      ! Initialize some pointers in OWN or GLOBAL memory to zeroes.
;  458          0587  2      !
;  459          0588  2      ! In V4 system, we can get the module end address from shareable image descriptor block.
;  460          0589  2      ! So, we don't have to guess what is the end address for the module.
;  461          0590  2      ! Note: Global symbols do not have static address table, this info. is needed for
;  462          0591  2      ! shareable image module static address table only.
;  463          0592  2      !
;  464          0593  2      IMAGE_MODULE_END_ADDR = 0;
;  465          0594  2      IMAGE_MODULE_RST_SIZE = 0;
;  466          0595  2      GST$BEGIN_ADDR = 0;
;  467          0596  2      GST$REC_COUNT = 0;
;  468          0597  2
;  469          0598  2
;  470          0599  2      ! Set the pointer to image header by using $CRMPSC to map in the 1st
;  471          0600  2      ! virtual block.
;  472          0601  2      !
;  473          0602  2      EXE_HDRBOUNDS[0] = 200;
;  474          0603  2      EXE_HDRBOUNDS[1] = 1^30;
;  475          0604  2      EXE_SECNAM_DESC[0] = 6;
;  476          0605  2      EXE_SECNAM_DESC[1] = UPLIT BYTE(%ASCIC 'HEADER');
;  477          0606  2      STATUS = $YS$CRMPSC(EXE_HDRBOUNDS, EXE_HDRBOUNDS, 0, SEC$M_EXPREG,
;  478          0607  2                          EXE_SECNAM_DESC, 0, 0,
;  479          0608  2                          .IMGFILCHAN, 1, 0, 0, 0);
;  480          0609  2      IF NOT .STATUS
;  481          0610  2      THEN
;  482          0611  3          BEGIN
;  483          0612  3          IMG_HEADER = 0;
;  484          0613  3          STATUS[STS$V_SEVERITY] = SYS$K_INFO;
;  485          0614  3          SIGNAL(.STATUS);
;  486          0615  3          RETURN FALSE;
;  487          0616  3          END
;  488          0617  3
;  489          0618  2      ELSE
;  490          0619  2          IMG_HEADER = .EXE_HDRBOUNDS[0];
;  491          0620  2
;  492          0621  2
;  493          0622  2      ! Start by doing some consistency checks to ensure that the image header
;  494          0623  2      ! pointer is non-zero and that the debug symbol-table offset in the header
;  495          0624  2      ! is non-zero.  If either is wrong, signal an error and give up.
;  496          0625  2      !
;  497          0626  2      IF .IMG_HEADER EQL 0
;  498          0627  2      THEN
;  499          0628  3          BEGIN
;  500          0629  3          SIGNAL(DBG$_NORSTBLD);
;  501          0630  3          RETURN FALSE;
;  502          0631  2          END;
;  503          0632  2
;  504          0633  2      IF .IMG_HEADER[IHD$W_SYMDBGOFF] EQL 0
;  505          0634  2      THEN
;  506          0635  3          BEGIN
;  507          0636  3          SIGNAL(DBG$_NORSTBLD);
;  508          0637  3          RETURN FALSE;
;  509          0638  2          END;
```

```
;  510          0639  2
;  511          0640  2
;  512          0641  2      ! We have a debug symbol table descriptor in the image header.  If no DST
;  513          0642  2      ! exists (because all the modules were compiled without /DEBUG or /TRACE),
;  514          0643  2      ! we indicate this by leaving the DST address zeroed.  Otherwise we set the
;  515          0644  2      ! DST address to 1 temporarily.
;  516          0645  2      !
;  517          0646  2      IMG_DST_BLOCK = .IMG_HEADER + .IMG_HEADER[IHD$W_SYMDBGOFF];
;  518          0647  2
;  519          0648  2
;  520          0649  2      ! First we want to map in the Global Symbol Table (GST) the same way.  First
;  521          0650  2      ! we make sure there is a GST--there must be at least four GST records (an
;  522          0651  2      ! image header, a sub-header, and an end-of-module record plus one data rec-
;  523          0652  2      ! ord).  If there is no GST, we leave the GST start address zeroed; other-
;  524          0653  2      ! wise we set it to 1 for now.
;  525          0654  2      !
;  526          0655  2      IF .IMG_DST_BLOCK[IHS$W_GSTRECS] GEQ 4 THEN GST$BEGIN_ADDR = 1;
;  527          0656  2
;  528          0657  2
;  529          0658  2      ! If the GST's Virtual Block Number (VBN) is invalid, we simply pretend that
;  530          0659  2      ! there is no GST, though this is actually an error in the image header.
;  531          0660  2      !
;  532          0661  2      IF .IMG_DST_BLOCK[IHS$L_GSTVBN] LEQ 2 THEN GST$BEGIN_ADDR = 0;
;  533          0662  2
;  534          0663  2
;  535          0664  2      ! We decided that there is a GST, so we create another map section
;  536          0665  2      ! which maps the GST into our virtual address space.  Again, if this fails,
;  537          0666  2      ! we print the status message with informational severity and set the GST
;  538          0667  2      ! address to zero.  If this succeeds, we make the GST pointer point to the
;  539          0668  2      ! first data record in the GST--we skip over the header and sub-header here.
;  540          0669  2      ! We also set the GST record count minus the header and trailer records.
;  541          0670  2      !
;  542          0671  2      IF .GST$BEGIN_ADDR NEQ 0
;  543          0672  2      THEN
;  544          0673  3          BEGIN
;  545          0674  3          EXE_SECBOUNDS[0] = 200;
;  546          0675  3          EXE_SECBOUNDS[1] = 1^30;
;  547          0676  3          EXE_SECNAM_DESC[0] = 3;
;  548          0677  3          EXE_SECNAM_DESC[1] = UPLIT BYTE(%ASCIC 'GST');
;  549          0678  3          STATUS = $VS$CRMPSC(EXE_SECBOUNDS, EXE_SECBOUNDS, 0, SEC$M_EXPREG,
;  550          0679  3                              EXE_SECNAM_DESC, 0, 0,
;  551          0680  3                              .IMGFILCHAN, .IMG_DST_BLOCK[IHS$W_GSTRECS],
;  552          0681  3                              .IMG_DST_BLOCK[IHS$L_GSTVBN], 0, 0);
;  553          0682  3          IF NOT .STATUS
;  554          0683  3          THEN
;  555          0684  4              BEGIN
;  556          0685  4              GST$BEGIN_ADDR = 0;
;  557          0686  4              STATUS[STS$V_SEVERITY] = SYS$K_INFO;
;  558          0687  4              SIGNAL(.STATUS);
;  559          0688  4              END
;  560          0689  4
;  561          0690  3          ELSE
;  562          0691  4              BEGIN
;  563          0692  4              GSTADDR = .EXE_SECBOUNDS[0];
;  564          0693  4              GSTADDR = .GSTADDR + 2 + ((.GSTADDR[0] + 1)/2)*2;
;  565          0694  4              GSTADDR = .GSTADDR + 2 + ((.GSTADDR[0] + 1)/2)*2;
;  566          0695  4              GST$BEGIN_ADDR = .GSTADDR;
```

```
;  567     0696  4            GST$NEXT_ADDR = .GST$BEGIN_ADDR;
;  568     0697  4            GST$REC_COUNT = .IMG_DST_BLOCK[IHS$W_GSTRECS] - 3;
;  569     0698  3            END;
;  570     0699  3
;  571     0700  2        END;
;  572     0701  2
;  573     0702  2
;  574     0703  2        ! If we have by now decided that there is no GST in the executable image,
;  575     0704  2        ! we return FALSE to the caller so that the caller print a message saying
;  576     0705  2        ! that there are no global symbols.
;  577     0706  2        !
;  578     0707  2        IF .GST$BEGIN_ADDR EQL 0 THEN RETURN FALSE;
;  579     0708  2
;  580     0709  2
;  581     0710  2        ! Set up the number of GST records to be read in.  Pick up the address of
;  582     0711  2        ! the current record, and update the pointer to the subsequent one.
;  583     0712  2        !
;  584     0713  2        GSTPTR = POSITION_GST(FALSE);
;  585     0714  2        IF .GSTPTR EQL 0 THEN SIGNAL(DBG$_INVGSTREC);
;  586     0715  2
;  587     0716  2
;  588     0717  2        ! Go through GST, build RST entries.
;  589     0718  2        !
;  590     0719  2        WHILE TRUE DO
;  591     0720  3            BEGIN
;  592     0721  3
;  593     0722  3
;  594     0723  3            ! Check to see if there are no more GST records.
;  595     0724  3            !
;  596     0725  3            IF .GSTPTR EQL 0 THEN EXITLOOP;
;  597     0726  3
;  598     0727  3
;  599     0728  3            ! See whether the current buffer address is beyond the end of the
;  600     0729  3            ! end of the last GST record we looked at.  Note that we rounded up
;  601     0730  3            ! GST$NEST_ADDR when calculating where the next GST record will begin.
;  602     0731  3            ! Therefore we must temporarily round it down again when comparing it
;  603     0732  3            ! with GSTPTR since it may point to the last unused byte in a GST
;  604     0733  3            ! record.
;  605     0734  3            !
;  606     0735  4            IF (.GSTPTR GEQA .GST$NEXT_ADDR - 1)
;  607     0736  3            THEN
;  608     0737  4                BEGIN
;  609     0738  5                IF ((GSTPTR = POSITION_GST(TRUE)) EQL 0)
;  610     0739  4                THEN
;  611     0740  4
;  612     0741  4
;  613     0742  4                    ! There are no more GST records.
;  614     0743  4                    !
;  615     0744  4                    EXITLOOP
;  616     0745  4
;  617     0746  4                ELSE
;  618     0747  4
;  619     0748  4
;  620     0749  4                    ! GET succeeded.  If the new record is a GST record, then
;  621     0750  4                    ! initialize the valiable GSTPTR to point to the first global
;  622     0751  4                    ! symbol definition block in this record.
;  623     0752  4                    !
```

```
;  624     0753  5                BEGIN
;  625     0754  5                LOCAL
;  626     0755  5                    BUFFER_ADDRESS: REF VECTOR[,BYTE];
;  627     0756  5
;  628     0757  5                BUFFER_ADDRESS = .GSTPTR;
;  629     0758  5                IF .BUFFER_ADDRESS[GST_RECORD_TYPE] EQL GST_TYPE
;  630     0759  5                THEN
;  631     0760  5                    GSTPTR = .GSTPTR + 1
;  632     0761  5                ELSE
;  633     0762  5
;  634     0763  5
;  635     0764  5                    ! This record is not a GST record. Go on to the next.
;  636     0765  5                    !
;  637     0766  5                    GSTPTR = %X'FFFFFFFF';
;  638     0767  5
;  639     0768  4                END;
;  640     0769  4
;  641     0770  4            END
;  642     0771  4
;  643     0772  4
;  644     0773  4        ! This is a global symbol.  Save its address.  Then update the
;  645     0774  4        ! GSTPTR to point to the next symbol.
;  646     0775  4        !
;  647     0776  3        ELSE
;  648     0777  4            BEGIN
;  649     0778  4            ! Do a CASE on the GST record type and handle each type separately.
;  650     0779  4            !
;  651     0780  4            CASE .GSTPTR[GSD$B_RECTYP] FROM GSD$C_PSC TO GSD$C_SPSC OF
;  652     0781  4                SET
;  653     0782  4
;  654     0783  4
;  655     0784  4                ! DEBUG does not handle PSECT, just update the address.
;  656     0785  4                !
;  657     0786  4                [GSD$C_PSC]:
;  658     0787  5                    BEGIN
;  659     0788  5                    GSTPTR = .GSTPTR + GPS$C_NAME + .GSTPTR[GPS$B_NAMLNG];
;  660     0789  4                    END;
;  661     0790  4
;  662     0791  4
;  663     0792  4                ! Handle the definition of a global symbol.  If this is the symbol
;  664     0793  4                ! we are looking for, build an RST entry for it and return a pointer
;  665     0794  4                ! to that RST entry.  Otherwise, continue the search.
;  666     0795  4                !
;  667     0796  4                [GSD$C_SYM]:
;  668     0797  5                    BEGIN
;  669     0798  5                    DBG$GST_TO_RST(.MODRSTPTR, GSTPTR[SDF$B_NAMLNG],.GSTPTR[SDF$L_VALUE] + .BASEADDR ,FALSE)
;  670     0799  5                    GSTPTR = .GSTPTR + SDF$C_NAME + .GSTPTR[SDF$B_NAMLNG];
;  671     0800  4                    END;
;  672     0801  4
;  673     0802  4
;  674     0803  4                ! Handle the definition of a global routine or a global entry point.
;  675     0804  4                ! If this is the symbol we are looking for, build an RST entry for
;  676     0805  4                ! it and return the entry's address.  Otherwise continue the search.
;  677     0806  4                !
;  678     0807  4                [GSD$C_EPM,
;  679     0808  4                 GSD$C_PRO]:
;  680     0809  5                    BEGIN
```

```
;  681          0810  5
;  682          0811  5
;  683          0812  5
;  684          0813  5
;  685          0814  5
;  686          0815  5
;  687          0816  5
;  688          0817  5
;  689          0818  5
;  690          0819  5
;  691          0820  5
;  692          0821  5
;  693          0822  5
;  694          0823  5
;  695          0824  6
;  696          0825  6
;  697          0826  5
;  698          0827  5
;  699          0828  5
;  700          0829  6
;  701          0830  6
;  702          0831  6
;  703          0832  6
;  704          0833  6
;  705          0834  7
;  706          0835  7
;  707          0836  7
;  708          0837  6
;  709          0838  6
;  710          0839  5
;  711          0840  5
;  712          0841  5
;  713          0842  5
;  714          0843  4
;  715          0844  4
;  716          0845  4
;  717          0846  5
;  718          0847  5
;  719          0848  5
;  720          0849  4
;  721          0850  4
;  722          0851  4
;  723          0852  4
;  724          0853  4
;  725          0854  4
;  726          0855  4
;  727          0856  4
;  728          0857  4
;  729          0858  5
;  730          0859  5
;  731          0860  5
;  732          0861  4
;  733          0862  4
;  734          0863  4
;  735          0864  4
;  736          0865  3
;  737          0866  3
```

```
! Keep track of the end address.  This is not needed for V4 system.
!
IF (.GSTPTR[PRO$L_ADDRS] + .BASEADDR) GTR .IMAGE_MODULE_END_ADDR
THEN
    IMAGE_MODULE_END_ADDR = .GSTPTR[PRO$L_ADDRS] + .BASEADDR;

DBG$GST_TO_RST(.MODRSTPTR, GSTPTR[PRO$B_NAMLNG],.GSTPTR[PRO$L_ADDRS] + .BASEADDR,TRUE);
CASE .GSTPTR[GSD$B_RECTYP] FROM GSD$C_EPM TO
                                GSD$C_PRO OF
    SET

    [GSD$C_EPM]:
        BEGIN
        GSTPTR = .GSTPTR + EPM$C_NAME + .GSTPTR[EPM$B_NAMLNG];
        END;

    [GSD$C_PRO]:
        BEGIN
        GSTPTR = .GSTPTR + PRO$C_NAME + .GSTPTR[PRO$B_NAMLNG];
        NUM_ARGS = .GSTPTR[FML$B_MAXARGS];
        GSTPTR = .GSTPTR + FML$C_SIZE;
        DECR I FROM .NUM_ARGS-1 TO 0 BY 1 DO
            BEGIN
            GSTPTR = .GSTPTR + .GSTPTR[ARG$B_BYTECNT] +
                ARG$C_SIZE;
            END;

            END;

                'ES;

        END;

    [GSD$C_SPSC]:
        BEGIN
        DBG$GST_TO_RST(.MODRSTPTR, GSTPTR[SGPS$B_NAMLNG],.GSTPTR[SGPS$L_BASE] + .BASEADDR,TRUE);
        GSTPTR = .GSTPTR + SGPS$C_NAME + .GSTPTR[SGPS$B_NAMLNG];
        END;


    ! We signal error for all other kinds.  There are other
    ! kinds of GST records, but we do not encounter them
    ! so far, if you do encounter them, see Linker man.
    ! (App. A) and Starlet.req.
    !
    [INRANGE, OUTRANGE]:
        BEGIN
        SIGNAL(DBG$_INVGSTTYP);
        EXITLOOP;
        END;

    TES;

END;                            ! End of ELSE a global symbol.
```

```
;  738     0867  2          END;                                ! End of WHILE loop over the GST.
;  739     0868  2
;  740     0869  2
;  741     0870  2      ! Release the Virtual address space occupied by Image header.
;  742     0871  2      !
;  743     0872  2      STATUS = SYS$DELTVA(EXE_HDRBOUNDS, EXE_HDRBOUNDS, 0);
;  744     0873  2      IF NOT .STATUS
;  745     0874  2      THEN
;  746     0875  3          BEGIN
;  747     0876  3          STATUS[STS$V_SEVERITY] = SYS$K_INFO;
;  748     0877  3          SIGNAL(.STATUS);
;  749     0878  2          END;
;  750     0879  2
;  751     0880  2      ! Release the Virtual address space occupied by GST.
;  752     0881  2      !
;  753     0882  2      STATUS = SYS$DELTVA(EXE_SECBOUNDS, EXE_SECBOUNDS, 0);
;  754     0883  2      IF NOT .STATUS
;  755     0884  2      THEN
;  756     0885  3          BEGIN
;  757     0886  3          STATUS[STS$V_SEVERITY] = SYS$K_INFO;
;  758     0887  3          SIGNAL(.STATUS);
;  759     0888  2          END;
;  760     0889  2
;  761     0890  2      ! Record the module size in the module RST.
;  762     0891  2      !
;  763     0892  2      MODRSTPTR[RST$L_MODRSTSIZ] = .IMAGE_MODULE_RST_SIZE * 4;
;  764     0893  2
;  765     0894  2      ! Fill in the module size in the program static address table.
;  766     0895  2      ! Needed only for V3 system.
;  767     0896  2      !
;  768     0897  2      IF NOT .DBG$GV_CONTROL[DBG$V_CONTROL_VERSION_4]
;  769     0898  2      THEN
;  770     0899  3          BEGIN
;  771     0900  3          IF .MODRSTPTR NEQ .RST$START_ADDR
;  772     0901  3          THEN
;  773     0902  4              BEGIN
;  774     0903  4              SATPTR = .SAT$START_ADDR;
;  775     0904  4              WHILE TRUE DO
;  776     0905  5                  BEGIN
;  777     0906  5                  IF .SATPTR EQL 0 THEN EXITLOOP;
;  778     0907  5                  IF .SATPTR[SAT$L_RSTPTR] EQL .MODRSTPTR
;  779     0908  5                  THEN
;  780     0909  6                      BEGIN
;  781     0910  6                      SATPTR[SAT$L_END] = .IMAGE_MODULE_END_ADDR;
;  782     0911  6                      EXITLOOP;
;  783     0912  5                      END;
;  784     0913  5
;  785     0914  5                  SATPTR = .SATPTR[SAT$L_FLINK];
;  786     0915  4                  END;
;  787     0916  4
;  788     0917  3              END;
;  789     0918  3
;  790     0919  2          END;
;  791     0920  2
;  792     0921  2
;  793     0922  2
;  794     0923  2
```

```
;  795          0924  2     RETURN TRUE;
;  796          0925  1     END;


                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                52  45  44  41  45  48  06  00050 P.AAE:  .ASCII  <6>\HEADER\                                    ;
                                            54  53  47  03  00057 P.AAF:  .ASCII  <3>\GST\                                       ;


                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                07FC 00000              .ENTRY  DBG$GST_INIT, Save R2,R3,R4,R5,R6,R7,R8,R9,-  ; 0525
                                                                                R10                                             ;
                                    5A      0000V   CF  9E 00002        MOVAB   DBG$GST_TO_RST, R10                             ;
                                    59 00000000G    00  9E 00007        MOVAB   SYS$DELTVA, R9                                  ;
                                    58 00000000G    00  9E 0000E        MOVAB   SYS$CRMPSC, R8                                  ;
                                    57 00000000'    EF  9E 00015        MOVAB   IMAGE_MODULE_END_ADDR, R7                       ;
                                    56 00000000G    00  9E 0001C        MOVAB   LIB$SIGNAL, R6                                  ;
                                    55 00000000'    EF  9E 00023        MOVAB   GST$BEGIN_ADDR, R5                              ;
                                    5E              1C  C2 0002A        SUBL2   #28, SP                                         ;
                                                FC  A7  7C 0002D        CLRQ    IMAGE_MODULE_RST_SIZE                           ; 0594
                                                65  D4 00030            CLRL    GST$BEGIN_ADDR                                  ; 0595
                                                08  A5  D4 00032        CLRL    GST$REC_COUNT                                   ; 0596
                                0C      AE      C8  8F  9A 00035        MOVZBL  #200, EXE_HDRBOUNDS                             ; 0602
                                10      AE 40000000 8F  D0 0003A        MOVL    #1073741824, EXE_HDRBOUNDS+4                    ; 0603
                                                06  DD 00042            PUSHL   #6                                              ; 0604
                                04      AE 00000000'    EF  9E 00044    MOVAB   P.AAE, EXE_SECNAM_DESC+4                        ; 0605
                                                7E  7C 0004C            CLRQ    -(SP)                                           ; 0606
                                    7E              01  7D 0004E        MOVQ    #1, -(SP)                                       ;
                                                04  AC  DD 00051        PUSHL   IMGFILCHAN                                      ; 0608
                                                7E  7C 00054            CLRQ    -(SP)                                           ; 0606
                                                1C  AE  9F 00056        PUSHAB  EXE_SECNAM_DESC                                 ;
                                        00020000    8F  DD 00059        PUSHL   #131072                                         ;
                                                7E  D4 0005F            CLRL    -(SP)                                           ;
                                                38  AE  9F 00061        PUSHAB  EXE_HDRBOUNDS                                   ;
                                                3C  AE  9F 00064        PUSHAB  EXE_HDRBOUNDS                                   ;
                                    68              0C  FB 00067        CALLS   #12, SYS$CRMPSC                                 ;
                                    53              50  D0 0006A        MOVL    R0, STATUS                                      ;
                                    08              53  E8 0006D        BLBS    STATUS, 1$                                      ; 0609
                                                    52  D4 00070        CLRL    IMG_HEADER                                      ; 0612
53              03                  00              03  F0 00072        INSV    #3, #0, #3, STATUS                              ; 0613
                                                    53  DD 00077        PUSHL   STATUS                                          ; 0614
                                                    11  11 00079        BRB     3$                                              ;
                                    52          10  AE  D0 0007B 1$:    MOVL    EXE_HDRBOUNDS, IMG_HEADER                       ; 0619
                                                    05  13 0007F        BEQL    2$                                              ; 0626
                                                04  A2  B5 00081        TSTW    4(IMG_HEADER)                                   ; 0633
                                                    0C  12 00084        BNEQ    4$                                              ;
                                        0002834A    8F  DD 00086 2$:    PUSHL   #164682                                         ; 0636
                                    66              01  FB 0008C 3$:    CALLS   #1, LIB$SIGNAL                                  ;
                                                0206    31 0008F        BRW     34$                                             ; 0637
                                    50          04  A2  3C 00092 4$:    MOVZWL  4(IMG_HEADER), R0                               ; 0646
                                    52              50  C0 00096        ADDL2   R0, IMG_DST_BLOCK                               ;
                                    04          0A  A2  B1 00099        CMPW    10(IMG_DST_BLOCK), #4                           ; 0655
                                                    03  1F 0009D        BLSSU   5$                                              ;
```

```
                65              01  D0 0009F          MOVL    #1, GST$BEGIN_ADDR
                02        04    A2  D1 000A2 5$:      CMPL    4(IMG_DST_BLOCK), #2                    ;  0661
                                02  14 000A6          BGTR    6$
                                65  D4 000A8          CLRL    GST$BEGIN_ADDR
                                65  D5 000AA 6$:      TSTL    GST$BEGIN_ADDR                          ;  0671
                                7C  13 000AC          BEQL    8$
          08    AE        C8    8F  9A 000AE          MOVZBL  #200, EXE_SECBOUNDS                     ;  0674
          0C    AE  40000000    8F  D0 000B3          MOVL    #1073741824, EXE_SECBOUNDS+4            ;  0675
                6E              03  D0 000BB          MOVL    #3, EXE_SECNAM_DESC                     ;  0676
          04    AE  00000000'   EF  9E 000BE          MOVAB   P.AAF, EXE_SECNAM_DESC+4                ;  0677
                                7E  7C 000C6          CLRQ    -(SP)                                   ;  0678
                          04    A2  DD 000C8          PUSHL   4(IMG_DST_BLOCK)                        ;  0681
                7E        0A    A2  3C 000CB          MOVZWL  10(IMG_DST_BLOCK), -(SP)                ;  0680
                          04    AC  DD 000CF          PUSHL   IMGFILCHAN
                                7E  7C 000D2          CLRQ    -(SP)                                   ;  0678
                          1C    AE  9F 000D4          PUSHAB  EXE_SECNAM_DESC
                    00020000    8F  DD 000D7          PUSHL   #131072
                                7E  D4 000DD          CLRL    -(SP)
                          30    AE  9F 000DF          PUSHAB  EXE_SECBOUNDS
                          34    AE  9F 000E2          PUSHAB  EXE_SECBOUNDS
                68              0C  FB 000E5          CALLS   #12, SYS$CRMPSC
                53              50  D0 000E8          MOVL    R0, STATUS
                0E              53  E8 000EB          BLBS    STATUS, 7$                              ;  0682
                                65  D4 000EE          CLRL    GST$BEGIN_ADDR                          ;  0685
          53          03        00  03 000F0          INSV    #3, #0, #3, STATUS                      ;  0686
                                53  DD 000F5          PUSHL   STATUS                                  ;  0687
                66              01  FB 000F7          CALLS   #1, LIB$SIGNAL
                                2E  11 000FA          BRB     8$                                      ;  0682
                51        08    AE  D0 000FC 7$:      MOVL    EXE_SECBOUNDS, GSTADDR                  ;  0692
                50              61  3C 00100          MOVZWL  (GSTADDR), R0                           ;  0693
                                50  D6 00103          INCL    R0
                50              02  C6 00105          DIVL2   #2, R0
                51        02 A140   3E 00108          MOVAW   2(GSTADDR)[R0], GSTADDR
                50              61  3C 0010D          MOVZWL  (GSTADDR), R0                           ;  0694
                                50  D6 00110          INCL    R0
                50              02  C6 00112          DIVL2   #2, R0
                51        02 A140   3E 00115          MOVAW   2(GSTADDR)[R0], GSTADDR
                65              51  D0 0011A          MOVL    GSTADDR, GST$BEGIN_ADDR                 ;  0695
          04    A5              65  D0 0011D          MOVL    GST$BEGIN_ADDR, GST$NEXT_ADDR           ;  0696
          08    A5        0A    A2  3C 00121          MOVZWL  10(IMG_DST_BLOCK), GST$REC_COUNT        ;  0697
          08    A5              03  C2 00126          SUBL2   #3, GST$REC_COUNT
                                65  D5 0012A 8$:      TSTL    GST$BEGIN_ADDR                          ;  0707
                                03  12 0012C          BNEQ    9$
                              0167  31 0012E          BRW     34$
                                7E  D4 00131 9$:      CLRL    -(SP)                                   ;  0713
        0000V   CF              01  FB 00133          CALLS   #1, POSITION_GST
                52              50  D0 00138          MOVL    R0, GSTPTR
                                09  12 0013B          BNEQ    10$                                     ;  0714
                    0002835A    8F  DD 0013D          PUSHL   #164698
                66              01  FB 00143          CALLS   #1, LIB$SIGNAL
                                52  D5 00146 10$:     TSTL    GSTPTR                                  ;  0725
                                4E  13 00148          BEQL    15$
    50          04  A5          01  C3 0014A          SUBL3   #1, GST$NEXT_ADDR, R0                   ;  0735
                50              52  D1 0014F          CMPL    GSTPTR, R0
                                1D  1F 00152          BLSSU   12$
                                01  DD 00154          PUSHL   #1                                      ;  0738
        0000V   CF              01  FB 00156          CALLS   #1, POSITION_GST
```

```
                                52              50  D0 0015B          MOVL    R0, GSTPTR                                      ;
                                                38  13 0015E          BEQL    15$                                             ;
                                50              52  D0 00160          MOVL    GSTPTR, BUFFER_ADDRESS                          ; 0757
                                01              60  91 00163          CMPB    (BUFFER_ADDRESS), #1                            ; 0758
                                                04  12 00166          BNEQ    11$                                             ;
                                                52  D6 00168          INCL    GSTPTR                                          ; 0760
                                                DA  11 0016A          BRB     10$                                             ;
                                52              01  CE 0016C 11$:     MNEGL   #1, GSTPTR                                      ; 0766
                                                D5  11 0016F          BRB     10$                                             ; 0735
                          0C                    62  8F 00171 12$:     CASEB   (GSTPTR), #0, #12                               ; 0780
                004D      004D          0031  0026     00175 13$:     .WORD   16$-13$,-                                       ;
                001A      001A          001A  001A     0017D                  18$-13$,-                                       ;
                001A      001A          001A  001A     00185                  20$-13$,-                                       ;
                                              009E     0018D                  20$-13$,-                                       ;
                                                                              14$-13$,-                                       ;
                                                                              14$-13$,-                                       ;
                                                                              14$-13$,-                                       ;
                                                                              14$-13$,-                                       ;
                                                                              14$-13$,-                                       ;
                                                                              14$-13$,-                                       ;
                                                                              14$-13$,-                                       ;
                                                                              14$-13$,-                                       ;
                                                                              27$-13$                                         ;
                                00028FB3        8F  DD 0018F 14$:     PUSHL   #167859                                         ; 0859
                                66              01  FB 00195          CALLS   #1, LIB$SIGNAL                                  ;
                                              0094  31 00198 15$:     BRW     28$                                             ; 0858
                                50        08    A2  9A 0019B 16$:     MOVZBL  8(GSTPTR), R0                                   ; 0788
                                52        09  A042  9E 0019F          MOVAB   9(R0)[GSTPTR], GSTPTR                           ;
                                                A0  11 001A4 17$:     BRB     10$                                             ; 0780
                                                7E  D4 001A6 18$:     CLRL    -(SP)                                           ; 0798
                      7E    05    A2        08    AC  C1 001A8          ADDL3   BASEADDR, 5(GSTPTR), -(SP)                      ;
                                              09    A2  9F 001AE          PUSHAB  9(GSTPTR)                                       ;
                                              0C    AC  DD 001B1          PUSHL   MODRSTPTR                                       ;
                                6A                  04  FB 001B4          CALLS   #4, DBG$GST_TO_RST                              ;
                                50        09    A2  9A 001B7          MOVZBL  9(GSTPTR), R0                                   ; 0799
                                52        0A  A042  9E 001BB          MOVAB   10(R0)[GSTPTR], GSTPTR                          ;
                                                84  11 001C0 19$:     BRB     10$                                             ; 0780
                50    05    A2        08    AC  C1 001C2 20$:     ADDL3   BASEADDR, 5(GSTPTR), R0                         ; 0814
                                67              50  D1 001C6          CMPL    R0, IMAGE_MODULE_END_ADDR                       ;
                                                03  15 001CB          BLEQ    21$                                             ;
                                67              50  D0 001CD          MOVL    R0, IMAGE_MODULE_END_ADDR                       ; 0816
                                                01  DD 001D0 21$:     PUSHL   #1                                              ; 0818
                                                50  DD 001D2          PUSHL   R0                                              ;
                                              0B    A2  9F 001D4          PUSHAB  11(GSTPTR)                                      ;
                                              0C    AC  DD 001D7          PUSHL   MODRSTPTR                                       ;
                                6A                  04  FB 001DA          CALLS   #4, DBG$GST_TO_RST                              ;
                01              02                  62  8F 001DD          CASEB   (GSTPTR), #2, #1                                ; 0819
                              000F                0004     001E1 22$:     .WORD   23$-22$,-                                       ;
                                                                              24$-22$                                         ;
                                50        0B    A2  9A 001E5 23$:     MOVZBL  11(GSTPTR), R0                                  ; 0825
                                52        0C  A042  9E 001E9          MOVAB   12(R0)[GSTPTR], GSTPTR                          ;
                                                B4  11 001EE          BRB     17$                                             ; 0819
                                50        0B    A2  9A 001F0 24$:     MOVZBL  11(GSTPTR), R0                                  ; 0830
                                52        0C  A042  9E 001F4          MOVAB   12(R0)[GSTPTR], GSTPTR                          ;
                                54        01    A2  90 001F9          MOVB    1(GSTPTR), NUM_ARGS                             ; 0831
                                52              02  C0 001FD          ADDL2   #2, GSTPTR                                      ; 0832
                                51              54  9A 00200          MOVZBL  NUM_ARGS, I                                     ; 0833
```

```
                                    09  11 00203             BRB     26$                                                     ;
                                50  01  A2 9A 00205 25$:      MOVZBL  1(GSTPTR), R0                                           ; 0835
                                52  02 A042 9E 00209          MOVAB   2(R0)[GSTPTR], GSTPTR                                   ;
                                F4      51 F4 0020C 26$:      SOBGEQ  I, 25$                                                  ; 0833
                                        91 11 00211           BRB     17$                                                     ; 0780
                                        01 DD 00213 27$:      PUSHL   #1                                                      ; 0847
                        7E      08  A2  08 AC C1 00215       ADDL3   BASEADDR, 8(GSTPTR), -(SP)                              ;
                                    A2  0C 9F 0021B           PUSHAB  12(GSTPTR)                                              ;
                                    AC  0C DD 0021E           PUSHL   MODRSTPTR                                               ;
                                6A      04 FB 00221           CALLS   #4, DBG$GST_TO_RST                                      ;
                                50  0C  A2 9A 00224           MOVZBL  12(GSTPTR), R0                                          ; 0848
                                52  0D A042 9E 00228          MOVAB   13(R0)[GSTPTR], GSTPTR                                  ;
                                        91 11 0022D           BRB     19$                                                     ; 0719
                                        7E D4 0022F 28$:      CLRL    -(SP)                                                   ; 0872
                                    14  AE 9F 00231           PUSHAB  EXE_HDRBOUNDS                                           ;
                                    18  AE 9F 00234           PUSHAB  EXE_HDRBOUNDS                                           ;
                                69      03 FB 00237           CALLS   #3, SYS$DELTVA                                          ;
                                53      50 D0 0023A           MOVL    R0, STATUS                                              ;
                                0A      53 E8 0023D           BLBS    STATUS, 29$                                             ; 0873
                    53  03      00      03 F0 00240           INSV    #3, #0, #3, STATUS                                      ; 0876
                                        53 DD 00245           PUSHL   STATUS                                                  ; 0877
                                66      01 FB 00247           CALLS   #1, LIB$SIGNAL                                          ;
                                        7E D4 0024A 29$:      CLRL    -(SP)                                                   ; 0883
                                    0C  AE 9F 0024C           PUSHAB  EXE_SECBOUNDS                                           ;
                                    10  AE 9F 0024F           PUSHAB  EXE_SECBOUNDS                                           ;
                                69      03 FB 00252           CALLS   #3, SYS$DELTVA                                          ;
                                53      50 D0 00255           MOVL    R0, STATUS                                              ;
                                0A      53 E8 00258           BLBS    STATUS, 30$                                             ; 0884
                    53  03      00      03 F0 0025B           INSV    #3, #0, #3, STATUS                                      ; 0887
                                        53 DD 00260           PUSHL   STATUS                                                  ; 0888
                                66      01 FB 00262           CALLS   #1, LIB$SIGNAL                                          ;
                                50      0C AC D0 00265 30$:   MOVL    MODRSTPTR, R0                                           ; 0894
                20  A0      FC  A7      02 78 00269           ASHL    #2, IMAGE_MODULE_RST_SIZE, 32(R0)                       ;
                    1D 00000000G    00      04 E0 0026F       BBS     #4, DBG$GV_CONTROL+1, 33$                               ; 0900
                            1C  A5      50 D1 00277           CMPL    R0, RST$START_ADDR                                      ; 0903
                                        17 13 0027B           BEQL    33$                                                     ;
                                51      24 A5 D0 0027D        MOVL    SAT$START_ADDR, SATPTR                                  ; 0906
                                        11 13 00281 31$:      BEQL    33$                                                     ; 0909
                                50      0C A1 D1 00283        CMPL    12(SATPTR), R0                                          ; 0910
                                        06 12 00287           BNEQ    32$                                                     ;
                                    08  A1      67 D0 00289   MOVL    IMAGE_MODULE_END_ADDR, 8(SATPTR)                        ; 0913
                                        05 11 0028D           BRB     33$                                                     ; 0912
                                51      61 D0 0028F 32$:      MOVL    (SATPTR), SATPTR                                        ; 0917
                                        ED 11 00292           BRB     31$                                                     ; 0907
                                50      01 D0 00294 33$:      MOVL    #1, R0                                                  ; 0924
                                        04 00297              RET                                                             ;
                                        50 D4 00298 34$:      CLRL    R0                                                      ; 0925
                                        04 0029A              RET                                                             ;

; Routine Size:  667 bytes,    Routine Base:  DBG$CODE + 02C8
```

```
;  798          0926  1 GLOBAL ROUTINE DBG$GST_TO_RST(MODRSTPTR, NAMEPTR, VALUE, ROUTINE_FLAG) :NOVALUE =
;  799          0927  1
;  800          0928  1 !++
;  801          0929  1 ! FUNCTION
;  802          0930  1 !       This routine accepts the characteristics of a global symbol as input
;  803          0931  1 !       and build a Run-Time Symbol Table (RST) entry for that symbol.
;  804          0932  1 !       The global symbol has normally been looked up in the Global Symbol
;  805          0933  1 !       Table (GST), and its name, value, and routine flag has been extracted
;  806          0934  1 !       from the GSf entry.  This routine is then called to convert that infor-
;  807          0935  1 !       mation to the corresponding RST pointer.  A new RST
;  808          0936  1 !       entry is constructed, added to the anonymous module's symbol chain,
;  809          0937  1 !       and added to the RST Hash Table.  In addition, a "dummy" DST entry is
;  810          0938  1 !       created for the symbol so that other symbol table access routines do
;  811          0939  1 !       not have to make a special case of global symbols--these symbols seem
;  812          0940  1 !       to have perfectly normal DST entries like all others.  The DST entry is
;  813          0941  1 !       created in the same memory block as the RST entry.
;  814          0942  1 !
;  815          0943  1 ! INPUTS
;  816          0944  1 !       MODRSTPTR - Module RST pointer.
;  817          0945  1 !
;  818          0946  1 !       NAMEPTR - A pointer to the global symbol's name, represented as a
;  819          0947  1 !                 Counted ASCII string.
;  820          0948  1 !
;  821          0949  1 !       VALUE   - The value (i.e., address) of the global symbol.
;  822          0950  1 !
;  823          0951  1 !       ROUTINE_FLAG - A flag set to TRUE if the global symbol is a routine
;  824          0952  1 !                 name.  If this flag is FALSE, the symbol is assumed to be
;  825          0953  1 !                 a data symbol.
;  826          0954  1 !
;  827          0955  1 ! OUTPUTS
;  828          0956  1 !       None.
;  829          0957  1 !
;  830          0958  2     BEGIN
;  831          0959  2
;  832          0960  2     MAP
;  833          0961  2         MODRSTPTR: REF RST$ENTRY,       ! Pointer to anonymous module RST entry
;  834          0962  2                                         !      or shareable image module
;  835          0963  2         NAMEPTR: REF VECTOR[,BYTE];     ! Pointer to Counted ASCII name of the
;  836          0964  2                                         !      global symbol
;  837          0965  2
;  838          0966  2     LOCAL
;  839          0967  2         DSTPTR: REF DST$RECORD,         ! Pointer to the dummy DST record we
;  840          0968  2                                         !      build for the global symbol
;  841          0969  2         LENGTH,                         ! Length of the RST entry
;  842          0970  2         RSTPTR: REF RST$ENTRY;          ! Pointer to the RST entry we construct
;  843          0971  2                                         !      for the global symbol
;  844          0972  2
;  845          0973  2
;  846          0974  2
;  847          0975  2     ! The global symbol is not yet in the RST.  If it is a routine name, con-
;  848          0976  2     ! struct a Routine RST Entry and a corresponding dummy DST record for it.
;  849          0977  2     !
;  850          0978  2     IF .ROUTINE_FLAG
;  851          0979  2     THEN
;  852          0980  3         BEGIN
;  853          0981  3         LENGTH = RST$K_ROUTENTSIZ + (11 + .NAMEPTR[0])/4;
;  854          0982  3         IMAGE_MODULE_RST_SIZE = .IMAGE_MODULE_RST_SIZE + .LENGTH;
```

```
;  855          0983  3          RSTPTR = GET_MEM(.LENGTH);
;  856          0984  3          DSTPTR = .RSTPTR + 4*RST$K_ROUTENTSIZ;
;  857          0985  3          RSTPTR[RST$B_KIND] = RST$K_ROUTINE;
;  858          0986  3          RSTPTR[RST$L_STARTADDR] = .VALUE;
;  859          0987  3          RSTPTR[RST$L_ENDADDR] = .VALUE;
;  860          0988  3          RSTPTR[RST$L_BREAKADDR] = .VALUE + 2;
;  861          0989  3          DSTPTR[DST$B_LENGTH] = 7 + .NAMEPTR[0];
;  862          0990  3          DSTPTR[DST$B_TYPE] = DST$K_RTNBEG;
;  863          0991  3          DSTPTR[DST$V_RTNBEG_UNUSED] = 0;
;  864          0992  3          DSTPTR[DST$V_RTNBEG_NO_CALL] = FALSE;
;  865          0993  3          DSTPTR[DST$L_RTNBEG_ADDRESS] = .VALUE;
;  866          0994  3          CH$MOVE(.NAMEPTR[0] + 1, NAMEPTR[0], DSTPTR[DST$B_RTNBEG_NAME]);
;  867          0995  3
;  868          0996  3
;  869          0997  3          ! One additional thing we do here is: If the global symbol
;  870          0998  3          ! we are adding is for the routine LIB$SIGNAL, then we
;  871          0999  3          ! save away its address. This is later used in DBGEVENT
;  872          1000  3          ! to discover whether we are stepping over a call to LIB$SIGNAL.
;  873          1001  3          ! The way we can get LIB$SIGNAL in the GST is if the user
;  874          1002  3          ! links /NOSYSSHR.
;  875          1003  3          !
;  876          1004  3          IF CH$EQL(11, .NAMEPTR, 11, UPLIT(%ASCIC 'LIB$SIGNAL'))
;  877          1005  3          THEN
;  878          1006  3              DBG$GL_LIB_SIGNAL_ADDR = .VALUE;
;  879          1007  3          IF CH$EQL(9, .NAMEPTR, 9, UPLIT(%ASCIC 'LIB$STOP'))
;  880          1008  3          THEN
;  881          1009  3              DBG$GL_LIB_STOP_ADDR = .VALUE;
;  882          1010  3          END
;  883          1011  3
;  884          1012  3
;  885          1013  3      ! If the global symbol is a data symbol, build a Data Item RST Entry for
;  886          1014  3      ! it.  Also construct the corresponding dummy DST record.  The DST data
;  887          1015  3      ! type is set to be longword integer.
;  888          1016  3      !
;  889          1017  2      ELSE
;  890          1018  3          BEGIN
;  891          1019  3          LENGTH = RST$K_DATENTSIZ + (11 + .NAMEPTR[0])/4;
;  892          1020  3          IMAGE_MODULE_RST_SIZE = .IMAGE_MODULE_RST_SIZE + .LENGTH;
;  893          1021  3          RSTPTR = GET_MEM(.LENGTH);
;  894          1022  3          DSTPTR = .RSTPTR + 4*RST$K_DATENTSIZ;
;  895          1023  3          RSTPTR[RST$B_KIND] = RST$K_DATA;
;  896          1024  3          RSTPTR[RST$L_TYPEPTR] = 0;
;  897          1025  3          DSTPTR[DST$B_LENGTH] = 7 + .NAMEPTR[0];
;  898          1026  3          DSTPTR[DST$B_TYPE] = DSC$K_DTYPE_L;
;  899          1027  3          DSTPTR[DST$B_VFLAGS] = 1;
;  900          1028  3          DSTPTR[DST$L_VALUE] = .VALUE;
;  901          1029  3          CH$MOVE(.NAMEPTR[0] + 1, NAMEPTR[0], DSTPTR[DST$B_NAME]);
;  902          1030  2          END;
;  903          1031  2
;  904          1032  2
;  905          1033  2      ! Fill in the remainder of the RST entry and mark the symbol as global.
;  906          1034  2      ! Attach the symbol to the anonymous module's symbol chain.
;  907          1035  2      !
;  908          1036  2      IF .MODRSTPTR EQL .RST$START_ADDR
;  909          1037  2      THEN
;  910          1038  2          RSTPTR[RST$V_GLOBAL] = TRUE
;  911          1039  2
```

```
;  912          1040  2
;  913          1041  2      ! Make a Static Address Table Entry.
;  914          1042  2      !
;  915          1043  2      ELSE
;  916          1044  3          BEGIN
;  917          1045  3          IMAGE_MODULE_RST_SIZE = .IMAGE_MODULE_RST_SIZE + SAT$K_ENTSIZE;
;  918          1046  3          STATIC_ADDRESS(.VALUE, 1, .RSTPTR, MODRSTPTR[RST$L_SAT_PTR], FALSE);
;  919          1047  2          END;
;  920          1048  2
;  921          1049  2      RSTPTR[RST$L_SYMCHNPTR] = .MODRSTPTR[RST$L_SYMCHNPTR];
;  922          1050  2      MODRSTPTR[RST$L_SYMCHNPTR] = .RSTPTR;
;  923          1051  2      RSTPTR[RST$L_DSTPTR] = .DSTPTR;
;  924          1052  2      RSTPTR[RST$L_UPSCOPEPTR] = .MODRSTPTR;
;  925          1053  2      RSTPTR[RST$W_REFCOUNT] = 0;
;  926          1054  2      DBG$HASH_INSERT(.RSTPTR);
;  927          1055  2
;  928          1056  2
;  929          1057  2      ! We are all done.
;  930          1058  2      !
;  931          1059  2      RETURN 0;
;  932          1060  2
;  933          1061  1      END;


                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                                 0005B          .BLKB   1
          00  4C  41  4E  47  49  53  24  42  49  4C  0A 0005C P.AAG:  .ASCII  <10>\LIB$SIGNAL\<0>                    ;
          00  00  00  50  4F  54  53  24  42  49  4C  08 00068 P.AAH:  .ASCII  <8>\LIB$STOP\<0><0><0>                 ;


                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                              0FFC 00000        .ENTRY  DBG$GST_TO_RST, Save R2,R3,R4,R5,R6,R7,R8,- ; 0926
                                                                R9,R10,R11
                    5B 00000000'  EF  9E 00002          MOVAB   IMAGE_MODULE_RST_SIZE, R11
                    58        08  AC  7D 00009          MOVQ    NAMEPTR, R8                                       ; 0981
                    6C        10  AC  E9 0000D          BLBC    ROUTINE_FLAG, 2$                                  ; 0978
                    50            68  9A 00011          MOVZBL  (R8), R0                                          ; 0981
                    50            0B  C0 00014          ADDL2   #11, R0
                    50            04  C6 00017          DIVL2   #4, R0
                    50            0B  C0 0001A          ADDL2   #11, LENGTH
                    6B            50  C0 0001D          ADDL2   LENGTH, IMAGE_MODULE_RST_SIZE                     ; 0982
                                  50  DD 00020          PUSHL   LENGTH                                            ; 0983
              0000V CF            01  FB 00022          CALLS   #1, GET_MEM
                    57            50  D0 00027          MOVL    R0, RSTPTR
                    56        2C  A7  9E 0002A          MOVAB   44(R7), DSTPTR                                    ; 0984
                    5A        14  A7  9E 0002E          MOVAB   20(R7), R10                                       ; 0985
                    6A            02  90 00032          MOVB    #2, (R10)
                 18 A7            59  D0 00035          MOVL    R9, 24(RSTPTR)                                    ; 0986
                 1C A7            59  D0 00039          MOVL    R9, 28(RSTPTR)                                    ; 0987
                 28 A7        02  A9  9E 0003D          MOVAB   2(R9), 40(RSTPTR)                                 ; 0988
              66                68  07  81 00042          ADDB3   #7, (R8), (DSTPTR)                                ; 0989
                 01 A6        BE  8F  9B 00046          MOVZBW  #190, 1(DSTPTR)                                   ; 0990
                 03 A6            59  D0 0004B          MOVL    R9, 3(DSTPTR)                                     ; 0993
```

```
                        50              68  9A 0004F        MOVZBL  (R8), R0                                ; 0994
                                        50  D6 00052        INCL    R0
                     07 A6              68  50 28 00054     MOVC3   R0, (R8), 7(DSTPTR)
          00000000' EF                  68  0B 29 00059     CMPC3   #11, (R8), P.AAG                        ; 1004
                                        07  12 00061        BNEQ    1$
                00000000G 00            59  D0 00063        MOVL    R9, DBG$GL_LIB_SIGNAL_ADDR              ; 1006
          00000000' EF                  68  09 29 0006A 1$: CMPC3   #9, (R8), P.AAR                        ; 1007
                                        48  12 00072        BNEQ    3$
                00000000G 00            59  D0 00074        MOVL    R9, DBG$GL_LIB_STOP_ADDR                ; 1009
                                        3F  11 0007B        BRB     3$                                      ; 0978
                        50              68  9A 0007D 2$:    MOVZBL  (R8), R0                                ; 1019
                        50              0B  C0 00080        ADDL2   #11, R0
                        50              04  C6 00083        DIVL2   #4, R0
                        50              07  C0 00086        ADDL2   #7, LENGTH
                        6B              50  C0 00089        ADDL2   LENGTH, IMAGE_MODULE_RST_SIZE           ; 1020
                                        50  DD 0008C        PUSHL   LENGTH                                  ; 1021
                    0000V CF            01  FB 0008E        CALLS   #1, GET_MEM
                        57              50  D0 00093        MOVL    R0, RSTPTR
                        56        1C    A7  9E 00096        MOVAB   28(R7), DSTPTR                          ; 1022
                        5A        14    A7  9E 0009A        MOVAB   20(R7), R10                             ; 1023
                        6A              06  90 0009E        MOVB    #6, (R10)
                                  18    A7  D4 000A1        CLRL    24(RSTPTR)                              ; 1024
                  66                    68  07 81 000A4     ADDB3   #7, (R8), (DSTPTR)                      ; 1025
                     01 A6        0108  8F  B0 000A8        MOVW    #264, 1(DSTPTR)                         ; 1026
                     03 A6              59  D0 000AE        MOVL    R9, 3(DSTPTR)                           ; 1028
                        50              68  9A 000B2        MOVZBL  (R8), R0                                ; 1029
                                        50  D6 000B5        INCL    R0
                  07 A6                 68  50 28 000B7     MOVC3   R0, (R8), 7(DSTPTR)
                        52        04    AC  D0 000BC 3$:    MOVL    MODRSTPTR, R2                           ; 1036
                00000000' EF            52  D1 000C0        CMPL    R2, RST$START_ADDR
                                        06  12 000C7        BNEQ    4$
                     01 AA              01  88 000C9        BISB2   #1, 1(R10)                              ; 1038
                                        13  11 000CD        BRB     5$
                        6B              05  C0 000CF 4$:    ADDL2   #5, IMAGE_MODULE_RST_SIZE               ; 1045
                                        7E  D4 000D2        CLRL    -(SP)                                   ; 1046
                                  18    A2  9F 000D4        PUSHAB  24(R2)
                                        57  DD 000D7        PUSHL   RSTPTR
                                        01  DD 000D9        PUSHL   #1
                                        59  DD 000DB        PUSHL   R9
                    0000V CF            05  FB 000DD        CALLS   #5, STATIC_ADDRESS
                     08 A7        08    A2  D0 000E2 5$:    MOVL    8(R2), 8(RSTPTR)                        ; 1049
                     08 A2              57  D0 000E7        MOVL    RSTPTR, 8(R2)                           ; 1050
                     0C A7              56  D0 000EB        MOVL    DSTPTR, 12(RSTPTR)                      ; 1051
                     10 A7              52  D0 000EF        MOVL    R2, 16(RSTPTR)                          ; 1052
                                  02    AA  B4 000F3        CLRW    2(R10)                                  ; 1053
                                        57  DD 000F6        PUSHL   RSTPTR                                  ; 1054
                    0000V CF            01  FB 000F8        CALLS   #1, DBG$HASH_INSERT
                                        04  000FD        RET                                             ; 1061

; Routine Size:  254 bytes,    Routine Base:  DBG$CODE + 0563
```

```
;  935          1062  1 GLOBAL ROUTINE DBG$HASH_FIND(NAMEPTR) =
;  936          1063  1 !
;  937          1064  1 ! FUNCTION
;  938          1065  1 !       This routine looks up a specified symbol name in the RST Hash Table and
;  939          1066  1 !       returns a pointer to the corresponding RST entry.  This routine is al-
;  940          1067  1 !       ways used in conjunction with routine DBG$HASH_FIND_SETUP.  Since there
;  941          1068  1 !       can in general be many RST entries with the same symbolic name, there
;  942          1069  1 !       has to be a way of finding all such entries.  DBG$HASH_FIND_SETUP is
;  943          1070  1 !       thus called to compute the initial hash index and to intialize this
;  944          1071  1 !       search.  DBG$HASH_FIND can then be called repeatedly to find each indi-
;  945          1072  1 !       vidual occurrence of the given name in the RST.  When no more RST
;  946          1073  1 !       entries with that name are found, a zero RST pointer is returned.
;  947          1074  1 !
;  948          1075  1 ! INPUTS
;  949          1076  1 !       NAMEPTR - A pointer to the counted string that gives the name of the
;  950          1077  1 !                 symbol to be looked up.  This name must be a single symbol
;  951          1078  1 !                 name, not a complete pathname.
;  952          1079  1 !
;  953          1080  1 !       The OWN variables DBG$HASH_RSTPTR and DBG$HASH_INDEX are implicit
;  954          1081  1 !                 inputs.  These variables are set by DBG$HASH_FIND_SETUP
;  955          1082  1 !                 initially.  Of these, variable DBG$HASH_RSTPTR is updated
;  956          1083  1 !                 by this routine.
;  957          1084  1 !
;  958          1085  1 ! OUTPUTS
;  959          1086  1 !       A pointer to the next RST entry with the specified symbol name is
;  960          1087  1 !                 returned as the routine's value.  If there are no more RST
;  961          1088  1 !                 entries with that name, zero is returned.
;  962          1089  1 !
;  963          1090  1 !       DBG$HASH_RSTPTR is an implicit output.  It is updated to contain the
;  964          1091  1 !                 returned RST pointer.
;  965          1092  1 !
;  966          1093  1
;  967          1094  2     BEGIN
;  968          1095  2
;  969          1096  2     MAP
;  970          1097  2         NAMEPTR: REF VECTOR[,BYTE];     ! A pointer to a counted string contain-
;  971          1098  2                                         !       ing the desired symbol name
;  972          1099  2
;  973          1100  2     LOCAL
;  974          1101  2         DSTNAMEPTR: REF VECTOR[,BYTE];  ! Pointer to DST name counted string
;  975          1102  2
;  976          1103  2
;  977          1104  2
;  978          1105  2     ! Loop through the current RST hash chain until the chain ends, i.e. until
;  979          1106  2     ! we get back to the chain list head.  For each RST entry, we see if it has
;  980          1107  2     ! the name we are searching for.  If it does, we return it to the caller.
;  981          1108  2     !
;  982          1109  2     IF .DBG$HASH_RSTPTR EQL 0 THEN $DBG_ERROR('RSTCNTRL\HASH_FIND');
;  983          1110  2     DBG$HASH_RSTPTR = .DBG$HASH_RSTPTR[RST$L_HASH_FLINK];
;  984          1111  2     WHILE .DBG$HASH_RSTPTR NEQ RST$HASH_TABLE[2*.DBG$HASH_INDEX] DO
;  985          1112  3         BEGIN
;  986          1113  3         IF .DBG$HASH_RSTPTR[RST$V_GLOBAL]
;  987          1114  3         THEN
;  988          1115  3             DBG$STA_SYMNAME(.DBG$HASH_RSTPTR, DSTNAMEPTR)
;  989          1116  3         ELSE
;  990          1117  3             DSTNAMEPTR = DBG$GET_DST_NAME(.DBG$HASH_RSTPTR[RST$L_DSTPTR]);
;  991          1118  3
```

```
;  992          1119  3          IF CH$EQL(.NAMEPTR[0], NAMEPTR[1], .DSTNAMEPTR[0], DSTNAMEPTR[1], 0)
;  993          1120  3          THEN
;  994          1121  3              RETURN .DBG$HASH_RSTPTR;
;  995          1122  3
;  996          1123  3          DBG$HASH_RSTPTR = .DBG$HASH_RSTPTR[RST$L_HASH_FLINK];
;  997          1124  2          END;
;  998          1125  2
;  999          1126  2
; 1000          1127  2      ! The hash chain ended before we found another RST symbol with the same
; 1001          1128  2      ! name.  Set DBG$HASH_RSTPTR to zero and return that as the routine value.
; 1002          1129  2      !
; 1003          1130  2      DBG$HASH_RSTPTR = 0;
; 1004          1131  2      RETURN 0;
; 1005          1132  2
; 1006          1133  1      END;


                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

5F 48 53 41 48 5C 4C 52 54 4E 43 54 53 52 12  00074 P.AAI:  .ASCII  <18>\RSTCNTRL\<92>\HASH_FIND\                ;
                                  44 4E 49 46  00083                                                            ;


                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                     003C 00000         .ENTRY  DBG$HASH_FIND, Save R2,R3,R4,R5                 ; 1062
                  55 00000000'   EF   9E 00002         MOVAB   DBG$HASH_RSTPTR, R5                             ;
                  5E             04   C2 00009         SUBL2   #4, SP                                          ;
                                 65   D5 0000C         TSTL    DBG$HASH_RSTPTR                                 ; 1109
                                 15   12 0000E         BNEQ    1$                                              ;
                     00000000'   EF   9F 00010         PUSHAB  P.AAI                                           ;
                                 01   DD 00016         PUSHL   #1                                              ;
                     00028362    8F   DD 00018         PUSHL   #164706                                         ;
                00000000G  00         03   FB 0001E    CALLS   #3, LIB$SIGNAL                                  ;
                           75         95   D0 00025 1$: MOVL    @DBG$HASH_RSTPTR, DBG$HASH_RSTPTR               ; 1110
                           54    04   AC   D0 00028      MOVL    NAMEPTR, R4                                     ; 1119
                           52         65   D0 0002C 2$: MOVL    DBG$HASH_RSTPTR, R2                             ; 1111
                     50    FC   A5   01   78 0002F      ASHL    #1, DBG$HASH_INDEX, R0                          ;
                           50  00A8 C540   DE 00034      MOVAL   RST$HASH_TAB[E[R0], R0                          ;
                           50         52   D1 0003A      CMPL    R2, R0                                          ;
                                      39   13 0003D      BEQL    6$                                              ;
                           0D    15   A2   E9 0003F      BLBC    21(R2), 3$                                      ; 1113
                              4004   8F   BB 00043      PUSHR   #^M<R2,SP>                                      ; 1115
                00000000G  00         02   FB 00047    CALLS   #2, DBG$STA_SYMNAME                             ;
                                      0B   11 0004E      BRB     4$                                              ;
                           0C    A2   DD 00050 3$:    PUSHL   12(R2)                                          ; 1117
                     F9B2  CF         01   FB 00053    CALLS   #1, DBG$GET_DST_NAME                            ;
                           6E         50   D0 00058    MOVL    R0, DSTNAMEPTR                                  ;
                           52    04   BC   9A 0005B 4$: MOVZBL  @NAMEPTR, R2                                   ; 1119
                           50         6E   D0 0005F      MOVL    DSTNAMEPTR, R0                                  ;
                           51         60   9A 00062      MOVZBL  (R0), R1                                        ;
51                   00    01   A4   52   2D 00065      CMPC5   R2, 1(R4), #0, R1, 1(R0)                        ;
                              01   A0        0006B                                                             ;
                                      04   12 0006D      BNEQ    5$                                              ;
                           50         65   D0 0006F      MOVL    DBG$HASH_RSTPTR, R0                             ; 1121
```

```
                    04 00072          RET                                                 ;
                75  95 D0 00073 5$:   MOVL    @DBG$HASH_RSTPTR, DBG$HASH_RSTPTR           ; 1123
                    B4 11 00076       BRB     2$                                          ; 1111
                    65 D4 00078 6$:   CLRL    DBG$HASH_RSTPTR                             ; 1130
                    50 D4 0007A       CLRL    R0                                          ; 1131
                    04 0007C          RET                                                 ; 1133

; Routine Size:  125 bytes,    Routine Base:  DBG$CODE + 0661
```

```
;1008     1134  1 GLOBAL ROUTINE DBG$HASH_FIND_SETUP(NAMEPTR): NOVALUE =
;1009     1135  1 !
;1010     1136  1 ! FUNCTION
;1011     1137  1 !       This routine sets up a search through the RST hash table for a specified
;1012     1138  1 !       symbol name.  The routine computes the symbol's hash index and saves it
;1013     1139  1 !       and a pointer to the corresponding RST hash chain's list head in OWN
;1014     1140  1 !       variables where they can be accessed by routine DBG$HASH_FIND.  After
;1015     1141  1 !       the initial DBG$HASH_FIND_SETUP call, DBG$HASH_FIND should be called
;1016     1142  1 !       repeatedly to find all RST entries with the specified symbol name.
;1017     1143  1 !
;1018     1144  1 !       Note that the hashing method is encoded both here and in routine
;1019     1145  1 !       DBG$HASH_INSERT.  If it is changed in one place, it must be changed
;1020     1146  1 !       in the other as well.
;1021     1147  1 !
;1022     1148  1 ! INPUTS
;1023     1149  1 !       NAMEPTR - A pointer to the counted string that gives the name of the
;1024     1150  1 !                 symbol to be looked up.  This name must be a single symbol
;1025     1151  1 !                 name, not a complete pathname.
;1026     1152  1 !
;1027     1153  1 ! OUTPUTS
;1028     1154  1 !       No value is returned by DBG$HASH_FIND_SETUP.
;1029     1155  1 !
;1030     1156  1 !       There are two implicit outputs: DBG$HASH_INDEX and DBG$HASH_RSTPTR
;1031     1157  1 !                 are set so they can be used by routine DBG$HASH_FIND.
;1032     1158  1 !
;1033     1159  1
;1034     1160  2     BEGIN
;1035     1161  2
;1036     1162  2     MAP
;1037     1163  2         NAMEPTR: REF VECTOR[,BYTE];     ! Pointer to symbol name counted string
;1038     1164  2
;1039     1165  2     LOCAL
;1040     1166  2         WORD_NAMEPTR: REF VECTOR[,WORD];! Pointer to symbol name counted string,
;1041     1167  2                                        ! used to access the string one word at
;1042     1168  2                                        ! a time.
;1043     1169  2
;1044     1170  2
;1045     1171  2     ! Compute the hash index for the specified symbol name.  Set DBG$HASH_INDEX
;1046     1172  2     ! and DBG$HASH_RSTPTR so they can be used by the DBG$HASH_FIND routine.
;1047     1173  2     !
;1048     1174  2     ! The hash algorithm is as follows:
;1049     1175  2     ! Add up the words in the name. (If the number of bytes is odd then include
;1050     1176  2     ! the count byte in the first word so it comes out even.) Then take
;1051     1177  2     ! that sum modulo the hash table size.
;1052     1178  2     !
;1053     1179  2     IF (.NAMEPTR[0] MOD 2) EQL 0
;1054     1180  2     THEN
;1055     1181  2         WORD_NAMEPTR = NAMEPTR[1]
;1056     1182  2     ELSE
;1057     1183  2         WORD_NAMEPTR = NAMEPTR[0];
;1058     1184  2
;1059     1185  2     DBG$HASH_INDEX = 0;
;1060     1186  2     INCR I FROM 0 TO (.NAMEPTR[0]+1)/2-1 DO
;1061     1187  2         DBG$HASH_INDEX = .DBG$HASH_INDEX + .WORD_NAMEPTR[.I];
;1062     1188  2
;1063     1189  2     DBG$HASH_INDEX = .DBG$HASH_INDEX MOD RST$K_HASHTBLSIZE;
;1064     1190  2     DBG$HASH_RSTPTR = RST$HASH_TABLE[2*.DBG$HASH_INDEX];
```

```
;  1065          1191  2     RETURN;
;  1066          1192  2
;  1067          1193  1     END;


                                                001C 00000          .ENTRY  DBG$HASH_FIND_SETUP, Save R2,R3,R4          ; 1134
                                    54 00000000'  EF   9E 00002     MOVAB   DBG$HASH_INDEX, R4
                                    50        04  BC   9A 00009     MOVZBL  @NAMEPTR, R0                                ; 1179
                7E              00  50            01   7A 0000D     EMUL    #1, R0, #0, -(SP)
                50              50  8E            02   7B 00012     EDIV    #2, (SP)+, R0, R0
                                                50   D5 00017     TSTL    R0
                                                07   12 00019     BNEQ    1$
                                52        04  AC            01   C1 0001B     ADDL3   #1, NAMEPTR, WORD_NAMEPTR           ; 1181
                                                04   11 00020     BRB     2$
                                52        04  AC            D0   00022 1$:  MOVL    NAMEPTR, WORD_NAMEPTR               ; 1183
                                                64   D4 00026 2$:  CLRL    DBG$HASH_INDEX                      ; 1185
                                51        04  BC            9A   00028     MOVZBL  @NAMEPTR, R1                        ; 1186
                                                51   D6 0002C     INCL    R1
                                51            02   C6 0002E     DIVL2   #2, R1
                                50            01   CE 00031     MNEGL   #1, I                                       ; 1187
                                                07   11 00034     BRB     4$
                                53          6240   3C 00036 3$:  MOVZWL  (WORD_NAMEPTR)[I], R3
                                64            53   C0 0003A     ADDL2   R3, DBG$HASH_INDEX
                            F5  50            51   F2 0003D 4$:  AOBLSS  R1, I, 3$
                7E              00  64            01   7A 00041     EMUL    #1, DBG$HASH_INDEX, #0, -(SP)               ; 1189
                50              50  8E  000009C7   8F   7B 00046     EDIV    #2503, (SP)+, R0, R0
                                64            50   D0 0004F     MOVL    R0, DBG$HASH_INDEX
                                50  64            01   78 00052     ASHL    #1, DBG$HASH_INDEX, R0                      ; 1190
                                    04  A4    00AC  C440   DE 00056     MOVAL   RST$HASH_TAB[E[R0], DBG$HASH_RSTPTR
                                                04   0005D     RET                                                 ; 1193

; Routine Size:  94 bytes,    Routine Base:  DBG$CODE + 06DE
```

```
;1069     1194  1 GLOBAL ROUTINE DBG$HASH_INSERT(RSTPTR): NOVALUE =
;1070     1195  1 !
;1071     1196  1 ! FUNCTION
;1072     1197  1 !       This routine inserts an RST entry in the RST Hash Table.  The symbolic
;1073     1198  1 !       name associated with the RST entry is looked up the DST and is hashed.
;1074     1199  1 !       The RST entry is then linked into the corresponding hash chain.
;1075     1200  1 !
;1076     1201  1 !       Note that the hashing method is encoded both here and in routine
;1077     1202  1 !       DBG$HASH_FIND_SETUP.  If it is changed in one place, it must thus
;1078     1203  1 !       be changed in the other as well.
;1079     1204  1 !
;1080     1205  1 ! INPUTS
;1081     1206  1 !       RSTPTR  - A pointer to the RST entry to be inserted in the hash table.
;1082     1207  1 !
;1083     1208  1 ! OUTPUTS
;1084     1209  1 !       NONE
;1085     1210  1 !
;1086     1211  1
;1087     1212  2     BEGIN
;1088     1213  2
;1089     1214  2     MAP
;1090     1215  2         RSTPTR: REF RST$ENTRY;          ! A pointer to the RST entry to be
;1091     1216  2                                         !      inserted in the RST Hash Table
;1092     1217  2
;1093     1218  2     LOCAL
;1094     1219  2         HASHINDEX,                      ! The computed hash index.
;1095     1220  2         NAMEPTR: REF VECTOR[,BYTE],     ! Pointer to symbol name (counted ASCII)
;1096     1221  2         NEXTPTR: REF RST$ENTRY,         ! A pointer to the RST entry in the hash
;1097     1222  2                                         !      chain before which the new RST
;1098     1223  2                                         !      entry is inserted.
;1099     1224  2         PREVPTR: REF RST$ENTRY,         ! A pointer to the RST entry in the hash
;1100     1225  2                                         !      chain after which the new RST
;1101     1226  2                                         !      entry is inserted.
;1102     1227  2         WORD_NAMEPTR: REF VECTOR[,WORD];! Pointer to symbol name counted string,
;1103     1228  2                                         ! used to access the string one word at
;1104     1229  2                                         ! a time.
;1105     1230  2
;1106     1231  2
;1107     1232  2
;1108     1233  2
;1109     1234  2     ! Get the RST entry's symbol name from the DST and compute its hash index.
;1110     1235  2     !
;1111     1236  2     IF .RSTPTR[RST$V_GLOBAL]
;1112     1237  2     THEN
;1113     1238  2         DBG$STA_SYMNAME(.RSTPTR, NAMEPTR)
;1114     1239  2     ELSE
;1115     1240  2         NAMEPTR = DBG$GET_DST_NAME(.RSTPTR[RST$L_DSTPTR]);
;1116     1241  2
;1117     1242  2
;1118     1243  2     ! Compute the hash index for the specified symbol name.  Set DBG$HASH_INDEX
;1119     1244  2     ! and DBG$HASH_RSTPTR so they can be used by the DBG$HASH_FIND routine.
;1120     1245  2     !
;1121     1246  2     ! The hash algorithm is as follows:
;1122     1247  2     ! Add up the words in the name. (If the number of bytes is odd then include
;1123     1248  2     ! the count byte in the first word so it comes out even.) Then take
;1124     1249  2     ! that sum modulo the hash table size.
;1125     1250  2     !
```

```
;  1126          1251  2     IF (.NAMEPTR[0] MOD 2) EQL 0
;  1127          1252  2     THEN
;  1128          1253  2         WORD_NAMEPTR = NAMEPTR[1]
;  1129          1254  2     ELSE
;  1130          1255  2         WORD_NAMEPTR = NAMEPTR[0];
;  1131          1256  2
;  1132          1257  2     HASHINDEX = 0;
;  1133          1258  2     INCR I FROM 0 TO (.NAMEPTR[0]+1)/2-1 DO
;  1134          1259  2         HASHINDEX = .HASHINDEX + .WORD_NAMEPTR[.I];
;  1135          1260  2
;  1136          1261  2     HASHINDEX = .HASHINDEX MOD RST$K_HASHTBLSIZE;
;  1137          1262  2
;  1138          1263  2
;  1139          1264  2     ! Link the RST entry into the appropriate hash chain.
;  1140          1265  2     !
;  1141          1266  2     PREVPTR = RST$HASH_TABLE[2*.HASHINDEX];
;  1142          1267  2     NEXTPTR = .PREVPTR[RST$L_HASH_FLINK];
;  1143          1268  2     RSTPTR[RST$L_HASH_FLINK] = .NEXTPTR;
;  1144          1269  2     RSTPTR[RST$L_HASH_BLINK] = .PREVPTR;
;  1145          1270  2     PREVPTR[RST$L_HASH_FLINK] = .RSTPTR;
;  1146          1271  2     NEXTPTR[RST$L_HASH_BLINK] = .RSTPTR;
;  1147          1272  2     RETURN;
;  1148          1273  2
;  1149          1274  1     END;

                                  003C 00000           .ENTRY  DBG$HASH_INSERT, Save R2,R3,R4,R5        ; 1194
                            5E          04   C2 00002           SUBL2   #4, SP
                            52    04    AC   D0 00005           MOVL    RSTPTR, R2                       ; 1236
                            0D    15    A2   E9 00009           BLBC    21(R2), 1$
                                4004    8F   BB 0000D           PUSHR   #^M<R2,SP>                       ; 1238
                  00000000G     00          02   FB 00011           CALLS   #2, DBG$STA_SYMNAME
                                            0B   11 00018           BRB     2$
                                      0C    A2   DD 0001A 1$:       PUSHL   12(R2)                       ; 1240
                        F90D    CF          01   FB 0001D           CALLS   #1, DBG$GET_DST_NAME
                            6E              50   D0 00022           MOVL    R0, NAMEPTR
                            50    00        BE   9A 00025 2$:       MOVZBL  @NAMEPTR, R0                 ; 1251
7E                          50              01   7A 00029           EMUL    #1, R0, #0, -(SP)
50                  50      8E              02   7B 0002E           EDIV    #2, (SP)+, R0, R0
                                            50   D5 00033           TSTL    R0
                                            06   12 00035           BNEQ    3$
                    54      6E              01   C1 00037           ADDL3   #1, NAMEPTR, WORD_NAMEPTR        ; 1253
                                            03   11 0003B           BRB     4$
                            54              6E   D0 0003D 3$:       MOVL    NAMEPTR, WORD_NAMEPTR            ; 1255
                                            50   D4 00040 4$:       CLRL    HASHINDEX                        ; 1257
                            53    00        BE   9A 00042           MOVZBL  @NAMEPTR, R3                     ; 1258
                                            53   D6 00046           INCL    R3
                            53              02   C6 00048           DIVL2   #2, R3
                            51              01   CE 0004B           MNEGL   #1, I                        ; 1259
                                            07   11 0004E           BRB     6$
                            55          6441    3C 00050 5$:       MOVZWL  (WORD_NAMEPTR)[I], R5
                            50              55   C0 00054           ADDL2   R5, HASHINDEX
                    F5      51              53   F2 00057 6$:       AOBLSS  R3, I, 5$
7E                  00      50              01   7A 0005B           EMUL    #1, HASHINDEX, #0, -(SP)         ; 1261
```

```
                50              50              8F 000009C7    8F   7B 00060         EDIV    #2503, (SP)+, HASHINDEX, HASHINDEX      ;
                                                50             02   C4 00069         MULL2   #2, R0                                  ; 1266
                                                51 00000000'EF 40   DE 0006C         MOVAL   RST$HASH_TABLE[R0], PREVPTR             ;
                                                50             61   D0 00074         MOVL    (PREVPTR), NEXTPTR                      ; 1267
                                                62             50   7D 00077         MOVQ    NEXTPTR, (R2)                           ; 1268
                                                61             52   D0 0007A         MOVL    R2, (PREVPTR)                           ; 1270
                                        04      A0             52   D0 0007D         MOVL    R2, 4(NEXTPTR)                          ; 1271
                                                                    04 00081         RET                                             ; 1274

; Routine Size:  130 bytes,    Routine Base:  DBG$CODE + 073C
```

```
;  1151     1275  1 GLOBAL ROUTINE DBG$RST_BUILD(MODRSTPTR): NOVALUE =
;  1152     1276  1 !
;  1153     1277  1 ! FUNCTION
;  1154     1278  1 !       This routine builds the RST for a specified module.  It takes as input a
;  1155     1279  1 !       pointer to the desired module's Module RST Entry.  From that it can get
;  1156     1280  1 !       to the module's DST records which are scanned until the Module-End DST
;  1157     1281  1 !       record is found.  For each DST record, a corresponding RST entry is cre-
;  1158     1282  1 !       ated if appropriate.  Each such RST entry is added to the module's sym-
;  1159     1283  1 !       bol chain and to the RST Hash Table.  Symbols with static addresses are
;  1160     1284  1 !       added to the module's Static Address Table (SAT) as well.
;  1161     1285  1 !
;  1162     1286  1 ! INPUTS
;  1163     1287  1 !       MODRSTPTR - A pointer to the Module RST Entry of the module whose RST
;  1164     1288  1 !                   is to be built.
;  1165     1289  1 !
;  1166     1290  1 ! OUTPUTS
;  1167     1291  1 !       The RST for the specified module is constructed.  No value is returned.
;  1168     1292  1 !
;  1169     1293  1
;  1170     1294  2     BEGIN
;  1171     1295  2
;  1172     1296  2     MAP
;  1173     1297  2         MODRSTPTR: REF RST$ENTRY;       ! Pointer to the Module RST Entry of the
;  1174     1298  2                                         !      module whose RST is to be built
;  1175     1299  2
;  1176     1300  2     LITERAL
;  1177     1301  2         MAX_NESTING     = 50;           ! Maximum lexical and record component
;  1178     1302  2                                         !      nesting we will allow
;  1179     1303  2
;  1180     1304  2     FIELD RS_FIELDS =                   ! Field definitions for RECSTACK array
;  1181     1305  2         SET                             !      to handle nested data records
;  1182     1306  2         RS_CLPTR        = [ 0, L_ ],    ! Component list pointer
;  1183     1307  2         RS_CLSIZ        = [ 1, L_ ],    ! Component list size
;  1184     1308  2         RS_RCDST        = [ 2, L_ ],    ! Record Begin DST address
;  1185     1309  2         RS_VRPTR        = [ 3, L_ ],    ! Variant record pointer
;  1186     1310  2         RS_TRSIZ        = [ 4, L_ ]     ! Type record size in longwords
;  1187     1311  2         TES;
;  1188     1312  2
;  1189     1313  2     LOCAL
;  1190     1314  2         ADDRESS,                        ! Static address to be added to SAT
;  1191     1315  2         ARRAY_FLAG,                     ! Flag set if item is array of records
;  1192     1316  2         BLINK: REF RST$ENTRY,           ! RST entry backward hash link
;  1193     1317  2         BUILD_TYPE_TABLE,               ! Flag saying whether we have to build
;  1194     1318  2                                         !      a type reference table.
;  1195     1319  2         COBOLGBL_FLAG,                  ! Flag set to TRUE is the COBOL "global"
;  1196     1320  2                                         !      attribute applies to the next
;  1197     1321  2                                         !      symbol defined in the DST
;  1198     1322  2         COMP_LST_PTR: REF VECTOR[,LONG],! Pointer to current component list
;  1199     1323  2         COMP_LST_SIZE,                  ! Size of current component list block
;  1200     1324  2         CONTINUATION_DST_SEEN:          ! Flag set when a Continuation DST
;  1201     1325  2                         INITIAL(FALSE), !      record is seen
;  1202     1326  2         DSTPTR: REF DST$RECORD,         ! Pointer to the current DST record
;  1203     1327  2         NEXT_DSTPTR: REF DST$RECORD,    ! Pointer to the next DST record
;  1204     1328  2         DSTPTR_TO_SAVE,                 ! The DST pointer to be saved in the
;  1205     1329  2                                         !      current RST entry
;  1206     1330  2         DST_TYPE,                       ! The DST type of the current DST record
;  1207     1331  2         ENUM_BITSIZE,                   ! Bit-size of current enumeration type
```

```
;  1208          1332  2          ENUMELT: REF VECTOR[,LONG],          ! Pointer to enumeration element list in
;  1209          1333  2                                                       the Type RST Entry
;  1210          1334  2          ENUM_DSTPTR,                         ! Pointer to Enumeration Type Begin DST
;  1211          1335  2                                                       record for current enum type
;  1212          1336  2          ENUM_LST_PTR: REF VECTOR[,LONG],     ! Pointer to list block used to accumu-
;  1213          1337  2                                                       late enumeration type elements
;  1214          1338  2          ENUM_LST_SIZE,                       ! Size of the ENUM_LST_PTR memory block
;  1215          1339  2          FLINK: REF RST$ENTRY,                ! RST entry forward hash link
;  1216          1340  2          LENGTH,                              ! Length of static address to add to SAT
;  1217          1341  2          MODNAMEPTR,                          ! Pointer to the module name as Counted
;  1218          1342  2                                                       ASCII--used for error messages
;  1219          1343  2          MOD_PC_TBL: REF VECTOR[,LONG],       ! Pointer to Module RST Entry vector of
;  1220          1344  2                                                       PC Correlation Table pointers
;  1221          1345  2          MOD_SRC_TBL: REF VECTOR[,LONG],      ! Pointer to Module RST Entry vector of
;  1222          1346  2                                                       pointers to module's Source Line
;  1223          1347  2                                                       Correlation DST Records
;  1224          1348  2          NAMEPTR,                             ! Pointer to symbol name counted ASCII
;  1225          1349  2          NEST_FLAG,                           ! Set to TRUE if the new RST entry def-
;  1226          1350  2                                                       ines a new level of nesting
;  1227          1351  2          NESTING,                             ! Current DST record nesting depth
;  1228          1352  2          NEST_STACK: VECTOR[MAX_NESTING],     ! Stack containing RST pointers for
;  1229          1353  2                                                       nested DST items: lexical enti-
;  1230          1354  2                                                       ties and record components
;  1231          1355  2          NESTING_BREAK,                       ! Current DST record break address
;  1232          1356  2                                                       nesting depth
;  1233          1357  2          NEST_BREAK_STACK: VECTOR[MAX_NESTING],
;  1234          1358  2                                               ! Stack containing RST pointers for
;  1235          1359  2                                                       nested DST items: Routine and
;  1236          1360  2                                                       Entry begin
;  1237          1361  2          NEWENT_FLAG,                         ! Flag set to TRUE if a new RST entry is
;  1238          1362  2                                                       being built from this DST record
;  1239          1363  2          NUM_PC_TBLS,                         ! Number of PC-Correlation Tables in the
;  1240          1364  2                                                       DST for the module being SET
;  1241          1365  2          NUM_SRC_DSTS,                        ! Number of Source Line Correlation DST
;  1242          1366  2                                                       Records for module being SET
;  1243          1367  2          PC_CORR_SIZE,                        ! Size of the PC_CORR_TBL memory block
;  1244          1368  2          PC_CORR_TBL: REF VECTOR[,LONG],       ! Pointer to table of pointers to the
;  1245          1369  2                                                       DST PC-Correlation Table records
;  1246          1370  2                                                       for the module being SET
;  1247          1371  2          PCTBL_BASE,                          ! Base address for offsets in current
;  1248          1372  2                                                       module's PC-Correlation Tables
;  1249          1373  2          PPTR: REF DST$PSECT_TRAILER,         ! Pointer to PSECT DST record trailer
;  1250          1374  2          RECBEGTRLR:                          ! Pointer to Record Begin DST record's
;  1251          1375  2                  REF DST$RECBEG_TRLR,                 trailer fields
;  1252          1376  2          REC_DSTPTR: REF DST$RECORD,          ! Pointer to Record Begin DST record
;  1253          1377  2          RECSTACK: BLOCKVECTOR                ! Record stack to keep track of nested
;  1254          1378  2                  [MAX_NESTING, 5]                     record and variant definitions
;  1255          1379  2                  FIELD(RS_FIELDS),
;  1256          1380  2          RSPTR,                               ! Current RECSTACK index (next available
;  1257          1381  2                                                       location in RECSTACK)
;  1258          1382  2          RSTPOINTER : REF RST$ENTRY,          ! Pointer to an RST entry
;  1259          1383  2          RSTPTR: REF RST$ENTRY,               ! Pointer to the RST entry we are
;  1260          1384  2                                                       currently building
;  1261          1385  2          RST_PTR: REF RST$ENTRY,              ! RST Entry pointer--used when fixing up
;  1262          1386  2                                                       pointers to continued DST records
;  1263          1387  2          SRC_CORR_SIZE,                       ! Size of the SRC_CORR_TBL memory block
;  1264          1388  2          SRC_CORR_TBL: REF VECTOR[,LONG],     ! Pointer to the table of pointers to
```

```
;  1265     1389  2                                    !       the Source Line Correlation DST
;  1266     1390  2                                    !       records of the module being SET
;  1267     1391  2          STATIC_FLAG,               ! Flag set to TRUE for static data items
;  1268     1392  2          SYMCHNPTR: REF RST$ENTRY,  ! Pointer to the last entry on the mod-
;  1269     1393  2                                    !       ule's symbol chain
;  1270     1394  2          TSPTR: REF DST$TYPE_SPEC,  ! Pointer to a type spec in a DST record
;  1271     1395  2          TAGDSTPTR,                 ! Address of the DST record of a record
;  1272     1396  2                                    !       variant's tag variable
;  1273     1397  2          TAGRSTPTR,                 ! Address of the RST entry of a record
;  1274     1398  2                                    !       variant's tag variable
;  1275     1399  2          TYPCOMPLST: REF VECTOR[,LONG],  ! Pointer to Type Component List
;  1276     1400  2          TYPE_DST_LST: REF VECTOR[,LONG],! Pointer to DST part of DST to RST
;  1277     1401  2                                    !       translate table for type entries
;  1278     1402  2          TYPE_DST_LEN,              ! Size of TYPE_RST_LST memory block
;  1279     1403  2          TYPEPTR: REF RST$ENTRY,    ! Pointer to Type RST Entry
;  1280     1404  2          TYPE_RST_LST: REF VECTOR[,LONG],! Pointer to RST part of DST to RST
;  1281     1405  2                                    !       translate table for type entries
;  1282     1406  2          TYPE_RST_LEN,              ! Size of TYPE_DST_LST memory block
;  1283     1407  2          TYPREFTBL: REF VECTOR[,LONG],  ! Pointer to Type RST Entry's type
;  1284     1408  2                                    !       reference table
;  1285     1409  2          TYP_RST_SIZE,              ! Size in longwords needed for the Type
;  1286     1410  2                                    !       RST memory block being built
;  1287     1411  2          UNNEST_FLAG,               ! Set to TRUE if this DST entry termi-
;  1288     1412  2                                    !       nates a level of DST nesting
;  1289     1413  2          VARBEG_DSTPTR:             ! Pointer to the Variant-Set Begin DST
;  1290     1414  2                  REF DST$VARBEG_TRAILER,!       record trailer fields
;  1291     1415  2          VARIANT_ENTRY:             ! Pointer to Variant RST Entry--used to
;  1292     1416  2                  REF RST$VAR_ENTRY, !       fix up pointers to continued DST
;  1293     1417  2                                    !       records
;  1294     1418  2          VARIANT_VALUE_FOUND,       ! Set to TRUE when a DST$K_VARVAL
;  1295     1419  2                                    !       record has been found
;  1296     1420  2          VAR_REC_PTR: REF RST$ENTRY,! Pointer to the current Variant RST
;  1297     1421  2                                    !       Entry being built
;  1298     1422  2          VARSETTBL: REF VECTOR[,LONG];  ! Pointer to Variant Set Table--used to
;  1299     1423  2                                    !       fix up pointers to continued DST
;  1300     1424  2                                    !       records
;  1301     1425  2
;  1302     1426  2      ENABLE
;  1303     1427  2          RST_BUILD_ERROR_HANDLER;   ! Set up error handler for DBG$_NOFREE
;  1304     1428  2
;  1305     1429  2
;  1306     1430  2
;  1307     1431  2      ! We have declared an error handler to handle the DBG$_NOFREE condition,
;  1308     1432  2      ! i.e. running out of memory while building the module's RST.  The handler
;  1309     1433  2      ! removes the partially built module from the RST.  We set the address of
;  1310     1434  2      ! the Module RST Entry in MOD_BEING_BUILT for the error handler to use.
;  1311     1435  2      ! This handler also handles the abort-on-control-Y condition.
;  1312     1436  2      !
;  1313     1437  2      MOD_BEING_BUILT = .MODRSTPTR;
;  1314     1438  2
;  1315     1439  2
;  1316     1440  2      ! If the module's RST has already been built, we return immediately.
;  1317     1441  2      ! Otherwise, we set the Module-in-RST flag and continue processing.
;  1318     1442  2      !
;  1319     1443  2      IF .MODRSTPTR[RST$B_KIND] NEQ RST$K_MODULE
;  1320     1444  2      THEN
;  1321     1445  2          $DBG_ERROR('RSTCNTRL\RST_BUILD 10');
```

```
;  1322          1446  2
;  1323          1447  2      IF .MODRSTPTR[RST$V_MOD_IN_RST] THEN RETURN;
;  1324          1448  2      MODRSTPTR[RST$V_MOD_IN_RST] = TRUE;
;  1325          1449  2
;  1326          1450  2
;  1327          1451  2      ! Mark this as the Most Recently Used module.  (The Most Recently Used mod-
;  1328          1452  2      ! ule will not be removed by DBG$GET_MEMORY to get more free space.)
;  1329          1453  2      !
;  1330          1454  2      DBG$RST_MOST_RECENT(.MODRSTPTR);
;  1331          1455  2
;  1332          1456  2
;  1333          1457  2      ! Build shareable image RST entries.
;  1334          1458  2      !
;  1335          1459  2      IF .MODRSTPTR[RST$V_SHARE_IMAGE]
;  1336          1460  2      THEN
;  1337          1461  3          BEGIN
;  1338          1462  3          IF NOT DBG$GST_INIT(.MODRSTPTR[RST$B_IMGFILCHAN], .MODRSTPTR[RST$L_BASEVA],
;  1339          1463  3                              .MODRSTPTR)
;  1340          1464  3          THEN
;  1341          1465  3              SIGNAL(DBG$_NOUNIVERSALS);
;  1342          1466  3
;  1343          1467  3          RETURN;
;  1344          1468  2          END;
;  1345          1469  2
;  1346          1470  2
;  1347          1471  2      ! Set the Debug Symbol Table (DST) pointer to point to the first DST record
;  1348          1472  2      ! after the specified module's Module-Begin record.  Note that we pick up
;  1349          1473  2      ! the language code from the Module Begin DST Record--unless the module has
;  1350          1474  2      ! been SET before, it may not be right in the Module RST Entry.
;  1351          1475  2      !
;  1352          1476  2      DSTPTR = .MODRSTPTR[RST$L_DSTPTR];
;  1353          1477  2      IF .DSTPTR[DST$B_TYPE] NEQ DST$K_MODBEG THEN SIGNAL(DBG$_INVDSTREC);
;  1354          1478  2      MODRSTPTR[RST$B_LANGUAGE] = .DSTPTR[DST$L_MODBEG_LANGUAGE];
;  1355          1479  2      MODNAMEPTR = DSTPTR[DST$B_MODBEG_NAME];
;  1356          1480  2      DSTPTR = DBG$RST_NEXT_DST( .DSTPTR );
;  1357          1481  2
;  1358          1482  2
;  1359          1483  2      ! If this is a PL/I module, we set the "old PL/I flag" to indicate that we
;  1360          1484  2      ! must compensate for the incorrect data descriptors generated by old ver-
;  1361          1485  2      ! sions of the PL/I compiler.  This flag is cleared if we find a Version
;  1362          1486  2      ! DST Record for this module later in the DST scan.
;  1363          1487  2      !
;  1364          1488  2      IF .MODRSTPTR[RST$B_LANGUAGE] EQL DBG$K_PLI
;  1365          1489  2      THEN
;  1366          1490  2          MODRSTPTR[RST$V_OLDPLIFLAG] = TRUE;
;  1367          1491  2
;  1368          1492  2
;  1369          1493  2      ! Initialize the module's Static Address Table pointer to zero.  Also set up
;  1370          1494  2      ! the module's symbol chain pointer.  Clear a random flag as well.
;  1371          1495  2      !
;  1372          1496  2      MODRSTPTR[RST$L_SAT_PTR] = 0;
;  1373          1497  2      SYMCHNPTR = .MODRSTPTR;
;  1374          1498  2      COBOLGBL_FLAG = FALSE;
;  1375          1499  2
;  1376          1500  2
;  1377          1501  2      ! Set up the two buffers we will use to accumulate pointers to the module's
;  1378          1502  2      ! PC-Correlation Table DST Records and its Source-Line Correlation DST
```

```
;  1379          1503  2     ! Records.  Initialize the number of such records to zero and set the PC-
;  1380          1504  2     ! Correlation Table base address to the largest possible address.
;  1381          1505  2     !
;  1382          1506  2     NUM_PC_TBLS = 0;
;  1383          1507  2     PC_CORR_SIZE = 100;
;  1384          1508  2     PC_CORR_TBL = GET_MEM(.PC_CORR_SIZE);
;  1385          1509  2     NUM_SRC_DSTS = 0;
;  1386          1510  2     SRC_CORR_SIZE = 100;
;  1387          1511  2     SRC_CORR_TBL = GET_MEM(.SRC_CORR_SIZE);
;  1388          1512  2     PCTBL_BASE = %X'7FFFFFFF';
;  1389          1513  2
;  1390          1514  2
;  1391          1515  2     ! Initialize the lexical nesting stack to have one entry pointing to the
;  1392          1516  2     ! Module RST Entry.
;  1393          1517  2     !
;  1394          1518  2     NESTING = 0;
;  1395          1519  2     NEST_STACK[0] = .MODRSTPTR;
;  1396          1520  2
;  1397          1521  2
;  1398          1522  2     ! Initialize the break address nesting stack.
;  1399          1523  2     !
;  1400          1524  2     NESTING_BREAK = 0;
;  1401          1525  2     NEST_BREAK_STACK[0] = 0;
;  1402          1526  2
;  1403          1527  2
;  1404          1528  2     ! Initialize the record and variant nesting stack to be empty and initialize
;  1405          1529  2     ! the variables which keep track of record and variant components.
;  1406          1530  2     !
;  1407          1531  2     RSPTR = 0;
;  1408          1532  2     COMP_LST_PTR = 0;
;  1409          1533  2     COMP_LST_SIZE = 0;
;  1410          1534  2     REC_DSTPTR = 0;
;  1411          1535  2     VAR_REC_PTR = 0;
;  1412          1536  2     TYP_RST_SIZE = 0;
;  1413          1537  2     ENUM_LST_PTR = 0;
;  1414          1538  2
;  1415          1539  2
;  1416          1540  2     ! Initialize the BUILD_TYPE_TABLE flag. This flag tells us whether we have
;  1417          1541  2     ! encountered any record types, for which we have to build a type
;  1418          1542  2     ! reference table. Avoiding the build of this table saves us some loops
;  1419          1543  2     ! over the RST at the end.
;  1420          1544  2     !
;  1421          1545  2     BUILD_TYPE_TABLE = FALSE;
;  1422          1546  2
;  1423          1547  2
;  1424          1548  2     ! Now loop over the whole DST for this module.  For each DST record we find,
;  1425          1549  2     ! we build the appropriate RST record and link all such records on the mod-
;  1426          1550  2     ! ule's symbol chain.  Each such symbol is also inserted in the RST Hash
;  1427          1551  2     ! Table.
;  1428          1552  2     !
;  1429          1553  2     WHILE TRUE DO
;  1430          1554  3         BEGIN
;  1431          1555  3
;  1432          1556  3
;  1433          1557  3         ! If the user entered Control-Y DEBUG, we signal our way out to abort
;  1434          1558  3         ! the current command.  The error handler does the necessary cleanup.
;  1435          1559  3         !
```

```
;  1436     1560  3          $ABORT_ON_CONTROL_Y;
;  1437     1561  3
;  1438     1562  3
;  1439     1563  3          ! Initialize flags for this DST record and get the DST record type.
;  1440     1564  3          !
;  1441     1565  3          NEWENT_FLAG = FALSE;
;  1442     1566  3          NEST_FLAG = FALSE;
;  1443     1567  3          UNNEST_FLAG = FALSE;
;  1444     1568  3          DST_TYPE = DST$K_MODBEG;
;  1445     1569  3          IF .DSTPTR LSSU .DST$END_ADDR
;  1446     1570  3          THEN
;  1447     1571  3              DST_TYPE = .DSTPTR[DST$B_TYPE];
;  1448     1572  3
;  1449     1573  3
;  1450     1574  3          ! Merge this DST record with its Continuation DST records (if any) and
;  1451     1575  3          ! set a flag if any continuation DST records have been seen.
;  1452     1576  3          !
;  1453     1577  3          IF MERGE_CONTINUATION_DST( DSTPTR, NEXT_DSTPTR )
;  1454     1578  3          THEN
;  1455     1579  3              CONTINUATION_DST_SEEN = TRUE;
;  1456     1580  3
;  1457     1581  3          DSTPTR_TO_SAVE = .DSTPTR;
;  1458     1582  3
;  1459     1583  3
;  1460     1584  3          ! Now handle each type of DST record individually.
;  1461     1585  3          !
;  1462     1586  3          CASE .DST_TYPE FROM 0 TO 255 OF
;  1463     1587  3              SET
;  1464     1588  3
;  1465     1589  3
;  1466     1590  3          ! Handle the Module-Begin record.  Since the Module-End record seems
;  1467     1591  3          ! to be missing, we signal an error and terminate the DST scan.
;  1468     1592  3          !
;  1469     1593  3          [DST$K_MODBEG]:
;  1470     1594  4              BEGIN
;  1471     1595  4              SIGNAL(DBG$_MISMODEND, 1, .MODNAMEPTR);
;  1472     1596  4              EXITLOOP;
;  1473     1597  4              END;
;  1474     1598  3
;  1475     1599  3
;  1476     1600  3          ! Handle the Module-End record.  Terminate the DST scan normally
;  1477     1601  3          ! after checking that the nesting level is down to 1.
;  1478     1602  3          !
;  1479     1603  3          [DST$K_MODEND]:
;  1480     1604  4              BEGIN
;  1481     1605  4              IF .NESTING NEQ 0 OR .NESTING_BREAK NEQ 0
;  1482     1606  4              THEN
;  1483     1607  4                  SIGNAL(DBG$_INCDSTNES, 1, .MODNAMEPTR);
;  1484     1608  4              EXITLOOP;
;  1485     1609  4              END;
;  1486     1610  3
;  1487     1611  3
;  1488     1612  3          ! Handle the Version record.  This record gives the version number
;  1489     1613  3          ! of the compiler that generated the current module.  If this is a
;  1490     1614  3          ! PL/I module and a Version DST record is present, we clear the
;  1491     1615  3          ! "old PL/I flag" in the Module RST Entry.  (This is the flag that
;  1492     1616  3          ! tells us to compensate for the incorrect symbol descriptors gene-
```

```
;  1493     1617  3              ! rated by old versions of the PL/I compiler.)
;  1494     1618  3              !
;  1495     1619  3              [DST$K_VERSION]:
;  1496     1620  4                  BEGIN
;  1497     1621  4                  IF .MODRSTPTR[RST$B_LANGUAGE] EQL DBG$K_PLI
;  1498     1622  4                  THEN
;  1499     1623  4                      MODRSTPTR[RST$V_OLDPLIFLAG] = FALSE;
;  1500     1624  4
;  1501     1625  3                  END;
;  1502     1626  3
;  1503     1627  3
;  1504     1628  3              ! Handle the PSECT record.  Enter it in the symbol table only if
;  1505     1629  3              ! the language is MACRO.  Also make a Static Address Table entry.
;  1506     1630  3              !
;  1507     1631  3              [DST$K_PSECT]:
;  1508     1632  4                  BEGIN
;  1509     1633  4                  IF .MODRSTPTR[RST$B_LANGUAGE] EQL DBG$K_MACRO
;  1510     1634  4                  THEN
;  1511     1635  5                      BEGIN
;  1512     1636  5
;  1513     1637  5
;  1514     1638  5                      ! See if there already is a label with the same name as the
;  1515     1639  5                      ! PSECT name (PSECTs are always declared last in MACRO).  If
;  1516     1640  5                      ! so, do not build a Label RST Entry for the PSECT.
;  1517     1641  5                      !
;  1518     1642  5                      NAMEPTR = DBG$GET_DST_NAME(.DSTPTR);
;  1519     1643  5                      DBG$HASH_FIND_SETUP(.NAMEPTR);
;  1520     1644  5                      WHILE TRUE DO
;  1521     1645  6                          BEGIN
;  1522     1646  6                          RSTPTR = DBG$HASH_FIND(.NAMEPTR);
;  1523     1647  6
;  1524     1648  6
;  1525     1649  6                      ! If there is no previous definition for the PSECT name,
;  1526     1650  6                      ! make a Label RST Entry for it.  Also make a Static
;  1527     1651  6                      ! Address Table entry.
;  1528     1652  6                      !
;  1529     1653  6                      IF .RSTPTR EQL 0
;  1530     1654  6                      THEN
;  1531     1655  7                          BEGIN
;  1532     1656  7                          RSTPTR = DBG$GET_MEMORY(RST$K_LBLENTSIZ);
;  1533     1657  7                          RSTPTR[RST$B_KIND] = RST$K_LABEL;
;  1534     1658  7                          RSTPTR[RST$L_STARTADDR] = .DSTPTR[DST$L_PSECT_VALUE];
;  1535     1659  7                          NEWENT_FLAG = TRUE;
;  1536     1660  7                          PPTR = DSTPTR[DST$A_PSECT_TRLR_BASE]
;  1537     1661  7                                 + .DSTPTR[DST$B_PSECT_TRLR_OFFS];
;  1538     1662  7                          LENGTH = .PPTR[DST$L_PSECT_SIZE];
;  1539     1663  7                          RSTPTR[RST$V_NONZLENGTH] = TRUE;
;  1540     1664  7                          STATIC_ADDRESS(.RSTPTR[RST$L_STARTADDR], .LENGTH,
;  1541     1665  7                              .RSTPTR, MODRSTPTR[RST$L_SAT_PTR], FALSE);
;  1542     1666  7                          EXITLOOP;
;  1543     1667  6                          END;
;  1544     1668  6
;  1545     1669  6
;  1546     1670  6                      ! If there already is a definition for the PSECT name in
;  1547     1671  6                      ! the current scope (i.e., the user defined a label by
;  1548     1672  6                      ! the same name), exit the name search loop without
;  1549     1673  6                      ! making a Label RST Entry for the PSECT.
```

```
;  1550     1674  6                !
;  1551     1675  6                IF .RSTPTR[RST$L_UPSCOPEPTR] EQL .NEST_STACK[.NESTING]
;  1552     1676  6                THEN
;  1553     1677  6                    EXITLOOP;
;  1554     1678  6
;  1555     1679  5                END;            ! End of Hash Table search loop
;  1556     1680  5
;  1557     1681  4            END;                ! End of language = MACRO if statement
;  1558     1682  4
;  1559     1683  3        END;
;  1560     1684  3
;  1561     1685  3
;  1562     1686  3        ! Handle the Line Number/PC Correlation Table record.  Here we save
;  1563     1687  3        ! its address in a temporary buffer.  It will later be moved to a
;  1564     1688  3        ! special table pointed to by the Module RST Entry.  Note that we
;  1565     1689  3        ! expand the buffer which holds the DST pointers if it overflows.
;  1566     1690  3        !
;  1567     1691  3        [DST$K_LINE_NUM,
;  1568     1692  3         DST$K_LINE_NUM_REL_R11]:
;  1569     1693  4            BEGIN
;  1570     1694  4            IF .NUM_PC_TBLS GEQ .PC_CORR_SIZE
;  1571     1695  4            THEN
;  1572     1696  5                BEGIN
;  1573     1697  5                MOD_PC_TBL = .PC_CORR_TBL;
;  1574     1698  5                PC_CORR_SIZE = .PC_CORR_SIZE + 100;
;  1575     1699  5                PC_CORR_TBL = DBG$GET_MEMORY(.PC_CORR_SIZE);
;  1576     1700  5                CH$MOVE(4*.NUM_PC_TBLS, .MOD_PC_TBL, .PC_CORR_TBL);
;  1577     1701  5                DBG$REL_MEMORY(.MOD_PC_TBL);
;  1578     1702  4                END;
;  1579     1703  4
;  1580     1704  4            PC_CORR_TBL[.NUM_PC_TBLS] = .DSTPTR;
;  1581     1705  4            NUM_PC_TBLS = .NUM_PC_TBLS + 1;
;  1582     1706  3            END;
;  1583     1707  3
;  1584     1708  3
;  1585     1709  3        ! Handle the Source Line Correlation record.  Here we save its
;  1586     1710  3        ! address in a temporary buffer.  It will later be moved into a
;  1587     1711  3        ! special table pointed to by the Module RST Entry.  Note that we
;  1588     1712  3        ! expand the buffer which holds the DST pointers if it overflows.
;  1589     1713  3        !
;  1590     1714  3        [DST$K_SOURCE]:
;  1591     1715  4            BEGIN
;  1592     1716  4            IF .NUM_SRC_DSTS GEQ .SRC_CORR_SIZE
;  1593     1717  4            THEN
;  1594     1718  5                BEGIN
;  1595     1719  5                MOD_SRC_TBL = .SRC_CORR_TBL;
;  1596     1720  5                SRC_CORR_SIZE = .SRC_CORR_SIZE + 100;
;  1597     1721  5                SRC_CORR_TBL = DBG$GET_MEMORY(.SRC_CORR_SIZE);
;  1598     1722  5                CH$MOVE(4*.NUM_SRC_DSTS, .MOD_SRC_TBL, .SRC_CORR_TBL);
;  1599     1723  5                DBG$REL_MEMORY(.MOD_SRC_TBL);
;  1600     1724  4                END;
;  1601     1725  4
;  1602     1726  4            SRC_CORR_TBL[.NUM_SRC_DSTS] = .DSTPTR;
;  1603     1727  4            NUM_SRC_DSTS = .NUM_SRC_DSTS + 1;
;  1604     1728  3            END;
;  1605     1729  3
;  1606     1730  3
```

```
;  1607     1731  3          ! Handle the Routine-Begin record.  Create the Routine RST Entry and
;  1608     1732  3          ! make a NEST_STACK entry so we know when the matching Routine-End
;  1609     1733  3          ! record is found (except that MACRO has no Routine-End records).
;  1610     1734  3          !
;  1611     1735  3          [DST$K_RTNBEG]:
;  1612     1736  4              BEGIN
;  1613     1737  4              RSTPTR = DBG$GET_MEMORY(RST$K_ROUTENTSIZ);
;  1614     1738  4              RSTPTR[RST$B_KIND] = RST$K_ROUTINE;
;  1615     1739  4              RSTPTR[RST$L_STARTADDR] = .DSTPTR[DST$L_RTNBEG_ADDRESS];
;  1616     1740  4              RSTPTR[RST$L_ENDADDR] = .RSTPTR[RST$L_STARTADDR];
;  1617     1741  4              IF .NESTING_BREAK GEQ MAX_NESTING
;  1618     1742  4              THEN
;  1619     1743  4                  SIGNAL(DBG$_DSTNESDEP, 1, .MODNAMEPTR);
;  1620     1744  4
;  1621     1745  4              ! Language MACRO does not have routine end dst record.
;  1622     1746  4              !
;  1623     1747  4              IF .MODRSTPTR[RST$B_LANGUAGE] NEQ DBG$K_MACRO
;  1624     1748  4              THEN
;  1625     1749  4                  NESTING_BREAK = .NESTING_BREAK + 1;
;  1626     1750  4
;  1627     1751  4              NEST_BREAK_STACK[.NESTING_BREAK] = .RSTPTR;
;  1628     1752  4              IF .DSTPTR[DST$V_RTNBEG_NO_CALL]
;  1629     1753  4              THEN
;  1630     1754  4                  RSTPTR[RST$L_BREAKADDR] = .RSTPTR[RST$L_STARTADDR]
;  1631     1755  4
;  1632     1756  4              ELSE
;  1633     1757  4                  RSTPTR[RST$L_BREAKADDR] = .RSTPTR[RST$L_STARTADDR] + 2;
;  1634     1758  4
;  1635     1759  4              NEWENT_FLAG = TRUE;
;  1636     1760  4              IF .PCTBL_BASE GTR .DSTPTR[DST$L_RTNBEG_ADDRESS]
;  1637     1761  4              THEN
;  1638     1762  4                  PCTBL_BASE = .DSTPTR[DST$L_RTNBEG_ADDRESS];
;  1639     1763  4
;  1640     1764  4              IF .MODRSTPTR[RST$B_LANGUAGE] EQL DBG$K_MACRO
;  1641     1765  4              THEN
;  1642     1766  4                  STATIC_ADDRESS(.RSTPTR[RST$L_STARTADDR], 0, .RSTPTR,
;  1643     1767  4                                 MODRSTPTR[RST$L_SAT_PTR], FALSE)
;  1644     1768  4              ELSE
;  1645     1769  4                  NEST_FLAG = TRUE;
;  1646     1770  4
;  1647     1771  3              END;
;  1648     1772  3
;  1649     1773  3
;  1650     1774  3          ! Handle the Routine-End record.  Compute the routine end address,
;  1651     1775  3          ! add it to the Routine RST Entry, make a Static Address Table
;  1652     1776  3          ! entry, and pop NEST_STACK.
;  1653     1777  3          !
;  1654     1778  3          [DST$K_RTNEND]:
;  1655     1779  4              BEGIN
;  1656     1780  4              NESTING_BREAK = .NESTING_BREAK - 1;
;  1657     1781  4              RSTPTR = .NEST_STACK[.NESTING];
;  1658     1782  4              IF .RSTPTR[RST$B_KIND] NEQ RST$K_ROUTINE
;  1659     1783  4              THEN
;  1660     1784  4                  SIGNAL(DBG$_INCDSTNES, 1, .MODNAMEPTR);
;  1661     1785  4
;  1662     1786  4              LENGTH = .DSTPTR[DST$L_RTNEND_SIZE];
;  1663     1787  4              IF .LENGTH LSS 1 THEN LENGTH = 1;
```

```
;  1664          1788  4                RSTPTR[RST$L_ENDADDR] = .RSTPTR[RST$L_STARTADDR] + .LENGTH - 1;
;  1665          1789  4                RSTPTR[RST$V_NONZLENGTH] = TRUE;
;  1666          1790  4                STATIC_ADDRESS(.RSTPTR[RST$L_STARTADDR], .LENGTH, .RSTPTR,
;  1667          1791  4                               MODRSTPTR[RST$L_SAT_PTR], FALSE);
;  1668          1792  4                UNNEST_FLAG = TRUE;
;  1669          1793  3                END;
;  1670          1794
;  1671          1795
;  1672          1796  3            ! Handle the Lexical Block-Begin record.  Create the Lexical Block
;  1673          1797  3            ! RST Entry and push the block on NEST_STACK.
;  1674          1798  3            !
;  1675          1799  3            [DST$K_BLKBEG]:
;  1676          1800  4                BEGIN
;  1677          1801  4                RSTPTR = DBG$GET_MEMORY(RST$K_LEXENTSIZ);
;  1678          1802  4                RSTPTR[RST$B_KIND] = RST$K_BLOCK;
;  1679          1803  4                RSTPTR[RST$L_STARTADDR] = .DSTPTR[DST$L_BLKBEG_ADDRESS];
;  1680          1804  4                RSTPTR[RST$L_ENDADDR] = .RSTPTR[RST$L_STARTADDR];
;  1681          1805  4                NEWENT_FLAG = TRUE;
;  1682          1806  4                NEST_FLAG = TRUE;
;  1683          1807  3                END;
;  1684          1808  3
;  1685          1809  3
;  1686          1810  3            ! Handle the Lexical Block-End record.  Compute the block's end ad-
;  1687          1811  3            ! dress, add it to the RST entry, make a Static Address Table entry,
;  1688          1812  3            ! and pop NEST_STACK.
;  1689          1813  3            !
;  1690          1814  3            [DST$K_BLKEND]:
;  1691          1815  4                BEGIN
;  1692          1816  4                RSTPTR = .NEST_STACK[.NESTING];
;  1693          1817  4                IF .RSTPTR[RST$B_KIND] NEQ RST$K_BLOCK
;  1694          1818  4                THEN
;  1695          1819  4                    SIGNAL(DBG$_INCDSTNES, 1, .MODNAMEPTR);
;  1696          1820  4
;  1697          1821  4                LENGTH = .DSTPTR[DST$L_BLKEND_SIZE];
;  1698          1822  4                IF .LENGTH LSS 1 THEN LENGTH = 1;
;  1699          1823  4                RSTPTR[RST$L_ENDADDR] = .RSTPTR[RST$L_STARTADDR] + .LENGTH - 1;
;  1700          1824  4                RSTPTR[RST$V_NONZLENGTH] = TRUE;
;  1701          1825  4                STATIC_ADDRESS(.RSTPTR[RST$L_STARTADDR], .LENGTH, .RSTPTR,
;  1702          1826  4                               MODRSTPTR[RST$L_SAT_PTR], FALSE);
;  1703          1827  4                UNNEST_FLAG = TRUE;
;  1704          1828  3                END;
;  1705          1829  3
;  1706          1830  3
;  1707          1831  3            ! Handle the Prolog record.  The Prolog DST record specifies the
;  1708          1832  3            ! address of the first instruction after the current routine's
;  1709          1833  3            ! prolog code, i.e. after the code needed to set up descriptors,
;  1710          1834  3            ! initialize the stack, and so on.  A routine breakpoint is set at
;  1711          1835  3            ! the prolog address for any routine that has a Prolog DST record.
;  1712          1836  3            !
;  1713          1837  3            [DST$K_PROLOG]:
;  1714          1838  4                BEGIN
;  1715          1839  4                RSTPTR = .NEST_BREAK_STACK[.NESTING_BREAK];
;  1716          1840  4                IF .RSTPTR NEQ 0
;  1717          1841  4                THEN
;  1718          1842  4                    RSTPTR[RST$L_BREAKADDR] = .DSTPTR[DST$L_PROLOG_BKPT_ADDR];
;  1719          1843  3                END;
;  1720          1844  3
```

```
;  1721     1845  3
;  1722     1846  3          ! Handle the Static Link record.  Just put a pointer to this DST
;  1723     1847  3          ! record in the Routine RST Entry currently on top of the nesting
;  1724     1848  3          ! stack.  If the top of the nesting stack is not a routine, signal
;  1725     1849  3          ! an error.
;  1726     1850  3          !
;  1727     1851  3          [DST$K_STATLINK]:
;  1728     1852  4              BEGIN
;  1729     1853  4              RSTPTR = .NEST_STACK[.NESTING];
;  1730     1854  4              IF .RSTPTR[RST$B_KIND] NEQ RST$K_ROUTINE
;  1731     1855  4              THEN
;  1732     1856  4                  SIGNAL(DBG$_INCDSTNES, 1, .MODNAMEPTR);
;  1733     1857  4
;  1734     1858  4              RSTPTR[RST$L_STATIC_LINK] = .DSTPTR;
;  1735     1859  3              END;
;  1736     1860  3
;  1737     1861  3
;  1738     1862  3          ! Handle the Label record.  Create a Label RST Entry for it and make
;  1739     1863  3          ! a Static Address Table entry.
;  1740     1864  3          !
;  1741     1865  3          [DST$K_LABEL]:
;  1742     1866  4              BEGIN
;  1743     1867  4              RSTPTR = BUILD_LABEL_RST(.DSTPTR);
;  1744     1868  4              DSTPTR_TO_SAVE = .RSTPTR[RST$L_DSTPTR];
;  1745     1869  4              RSTPTR[RST$B_KIND] = RST$K_LABEL;
;  1746     1870  4              RSTPTR[RST$L_STARTADDR] = .DSTPTR[DST$L_VALUE];
;  1747     1871  4              NEWENT_FLAG = TRUE;
;  1748     1872  4              STATIC_ADDRESS(.RSTPTR[RST$L_STARTADDR], 0, .RSTPTR,
;  1749     1873  4                                  MODRSTPTR[RST$L_SAT_PTR], FALSE);
;  1750     1874  3              END;
;  1751     1875  3
;  1752     1876  3
;  1753     1877  3          ! Handle the Label-or-Literal record.  If this is an instruction
;  1754     1878  3          ! label, create a Label RST Entry and a Static Address Table entry.
;  1755     1879  3          ! If this is a literal, create a Data Item RST Entry for it.
;  1756     1880  3          !
;  1757     1881  3          [DST$K_LBLORLIT]:
;  1758     1882  4              BEGIN
;  1759     1883  4              IF .DSTPTR[DST$B_VFLAGS] EQL 1
;  1760     1884  4              THEN
;  1761     1885  5                  BEGIN
;  1762     1886  5                  RSTPTR = BUILD_LABEL_RST(.DSTPTR);
;  1763     1887  5                  DSTPTR_TO_SAVE = .RSTPTR[RST$L_DSTPTR];
;  1764     1888  5                  RSTPTR[RST$B_KIND] = RST$K_LABEL;
;  1765     1889  5                  RSTPTR[RST$L_STARTADDR] = .DSTPTR[DST$L_VALUE];
;  1766     1890  5                  STATIC_ADDRESS(.RSTPTR[RST$L_STARTADDR], 0, .RSTPTR,
;  1767     1891  5                                  MODRSTPTR[RST$L_SAT_PTR], FALSE);
;  1768     1892  5                  END
;  1769     1893  5
;  1770     1894  4              ELSE
;  1771     1895  5                  BEGIN
;  1772     1896  5                  RSTPTR = DBG$GET_MEMORY(RST$K_DATENTSIZ);
;  1773     1897  5                  RSTPTR[RST$B_KIND] = RST$K_DATA;
;  1774     1898  5                  RSTPTR[RST$L_TYPEPTR] = 0;
;  1775     1899  4                  END;
;  1776     1900  4
;  1777     1901  4              NEWENT_FLAG = TRUE;
```

```
;  1778          1902  3              END;
;  1779          1903  3
;  1780          1904  3
;  1781          1905  3          ! Handle the Entry-Point record.  Create an Entry-Point RST Entry
;  1782          1906  3          ! for it and make a Static Address Table entry.
;  1783          1907  3          !
;  1784          1908  3          [DST$K_ENTRY]:
;  1785          1909  4              BEGIN
;  1786          1910  4              RSTPTR = DBG$GET_MEMORY(RST$K_EPTENTSIZ);
;  1787          1911  4              RSTPTR[RST$B_KIND] = RST$K_ENTRY;
;  1788          1912  4              RSTPTR[RST$L_STARTADDR] = .DSTPTR[DST$L_VALUE];
;  1789          1913  4              NEST_BREAK_STACK[.NESTING_BREAK] = .RSTPTR;
;  1790          1914  4              RSTPTR[RST$L_BREAKADDR] = .RSTPTR[RST$L_STARTADDR] + 2;
;  1791          1915  4              NEWENT_FLAG = TRUE;
;  1792          1916  4              STATIC_ADDRESS(.RSTPTR[RST$L_STARTADDR], 0, .RSTPTR,
;  1793          1917  4                                  MODRSTPTR[RST$L_SAT_PTR], FALSE);
;  1794          1918  3              END;
;  1795          1919  3
;  1796          1920  3
;  1797          1921  3          ! Handle the Record-Begin record.  Generate a Data Item RST Entry
;  1798          1922  3          ! for the record unless this is just a Type definition.  Then stack
;  1799          1923  3          ! the information which keeps track of the currently open record
;  1800          1924  3          ! definition (if any) and set up the same information for this new
;  1801          1925  3          ! record.
;  1802          1926  3          !
;  1803          1927  3          [DST$K_RECBEG]:
;  1804          1928  4              BEGIN
;  1805          1929  4
;  1806          1930  4              ! Set the flag saying we have to build the Type Reference
;  1807          1931  4              ! Table. We always have to build this table in the presence
;  1808          1932  4              ! of records.
;  1809          1933  4              !
;  1810          1934  4              BUILD_TYPE_TABLE = TRUE;
;  1811          1935  4
;  1812          1936  4
;  1813          1937  4              ! Generate a Data Item RST Entry if this is a data record defin-
;  1814          1938  4              ! ition (not just a Type definition).
;  1815          1939  4              !
;  1816          1940  4              IF .DSTPTR[DST$B_VFLAGS] NEQ DST$K_VFLAGS_NOVAL
;  1817          1941  4              THEN
;  1818          1942  5                  BEGIN
;  1819          1943  5                  RSTPTR = DBG$GET_MEMORY(RST$K_DATENTSIZ);
;  1820          1944  5                  RSTPTR[RST$B_KIND] = RST$K_DATA;
;  1821          1945  5                  RSTPTR[RST$L_TYPEPTR] = .DSTPTR;
;  1822          1946  5                  RSTPTR[RST$V_SET_TYPEPTR] = TRUE;
;  1823          1947  5                  NEWENT_FLAG = TRUE;
;  1824          1948  5                  IF .COMP_LST_PTR EQL 0
;  1825          1949  5                  THEN
;  1826          1950  6                      BEGIN
;  1827          1951  6                      STATIC_CHECK(.DSTPTR, ADDRESS, LENGTH, STATIC_FLAG);
;  1828          1952  6                      IF .STATIC_FLAG
;  1829          1953  6                      THEN
;  1830          1954  7                          BEGIN
;  1831          1955  7                          IF .LENGTH NEQ 0 THEN RSTPTR[RST$V_NONZLENGTH] = TRUE;
;  1832          1956  7                          STATIC_ADDRESS(.ADDRESS, .LENGTH, .RSTPTR,
;  1833          1957  7                                          MODRSTPTR[RST$L_SAT_PTR], FALSE);
;  1834          1958  6                          END;
```

```
;  1835          1959  6                        END
;  1836          1960  6
;  1837          1961  6                    ELSE
;  1838          1962  5                        COMP_LIST_ADD_ENTRY(COMP_LST_PTR,COMP_LST_SIZE,.RSTPTR);
;  1839          1963  5
;  1840          1964  5                    END;
;  1841          1965  4
;  1842          1966  4
;  1843          1967  4                    ! Stack the variables which keep track of the current record
;  1844          1968  4                    ! definition (if any).  Then set these variables up to keep
;  1845          1969  4                    ! track of this new record definition.
;  1846          1970  4                    !
;  1847          1971  4                    IF .RSPTR GEQ MAX_NESTING
;  1848          1972  4                    THEN
;  1849          1973  4                        SIGNAL(DBG$_DSTNESDEP, 1, .MODNAMEPTR);
;  1850          1974  4
;  1851          1975  4                    RECSTACK[.RSPTR, RS_CLPTR] = .COMP_LST_PTR;
;  1852          1976  4                    RECSTACK[.RSPTR, RS_CLSIZ] = .COMP_LST_SIZE;
;  1853          1977  4                    RECSTACK[.RSPTR, RS_RCDST] = .REC_DSTPTR;
;  1854          1978  4                    RECSTACK[.RSPTR, RS_VRPTR] = .VAR_REC_PTR;
;  1855          1979  4                    RECSTACK[.RSPTR, RS_TRSIZ] = .TYP_RST_SIZE;
;  1856          1980  4                    RSPTR = .RSPTR + 1;
;  1857          1981  4                    COMP_LST_PTR = DBG$GET_MEMORY(11);
;  1858          1982  4                    COMP_LST_SIZE = 10;
;  1859          1983  4                    REC_DSTPTR = .DSTPTR;
;  1860          1984  4                    VAR_REC_PTR = 0;
;  1861          1985  4                    TYP_RST_SIZE = RST$K_TYPENTSIZ;
;  1862          1986  4                    END;
;  1863          1987  3
;  1864          1988  3
;  1865          1989  3
;  1866          1990  3                ! Handle the Record-End record.  Here we create the Data Type RST
;  1867          1991  3                ! Entry for the record type.  We then pop the record stack to re-
;  1868          1992  3                ! store the context for any outer record definition.
;  1869          1993  3                !
;  1870          1994  3                [DST$K_RECEND]:
;  1871          1995  4                    BEGIN
;  1872          1996  4                    IF .VAR_REC_PTR NEQ 0
;  1873          1997  4                    THEN
;  1874          1998  4                        SIGNAL(DBG$_INCDSTNES, 1, .MODNAMEPTR);
;  1875          1999  4
;  1876          2000  4
;  1877          2001  4                    ! Build the Data Type RST Entry for the record type.
;  1878          2002  4                    !
;  1879          2003  4                    TYP_RST_SIZE = .TYP_RST_SIZE + .COMP_LST_PTR[0];
;  1880          2004  4                    RSTPTR = DBG$GET_MEMORY(.TYP_RST_SIZE);
;  1881          2005  4                    RSTPTR[RST$B_KIND] = RST$K_TYPE;
;  1882          2006  4                    RSTPTR[RST$B_FCODE] = RST$K_TYPE_RECORD;
;  1883          2007  4                    RECBEGTRLR = REC_DSTPTR[DST$B_NAME] + 1 + .REC_DSTPTR[DST$B_NAME];
;  1884          2008  4                    RSTPTR[RST$L_BITSIZE] = .RECBEGTRLR[DST$L_RECBEG_SIZE];
;  1885          2009  4                    DSTPTR_TO_SAVE = .REC_DSTPTR;
;  1886          2010  4                    NEWENT_FLAG = TRUE;
;  1887          2011  4
;  1888          2012  4
;  1889          2013  4                    ! Now copy all component and variant set lists into the Type RST
;  1890          2014  4                    ! Entry's memory block after the RST entry proper.  This is done
;  1891          2015  4                    ! by a call on recursive routine COPY_COMP_LIST.
```

```
;  1892          2016  4                 !
;  1893          2017  4                 TYPRSTPTR = .RSTPTR;
;  1894          2018  4                 TYPRSTLOC = RSTPTR[RST$L_TYPCOMPCNT];
;  1895          2019  4                 COPY_COMP_LIST(.COMP_LST_PTR);
;  1896          2020  4
;  1897          2021  4
;  1898          2022  4                 ! Pop the record stack so we continue the accumulation of record
;  1899          2023  4                 ! components at the next outer level of record nesting (if any).
;  1900          2024  4                 !
;  1901          2025  4                 IF .RSPTR LEQ 0 THEN SIGNAL(DBG$_INCDSTNES, 1, .MODNAMEPTR);
;  1902          2026  4                 RSPTR = .RSPTR - 1;
;  1903          2027  4                 COMP_LST_PTR = .RECSTACK[.RSPTR, RS_CLPTR];
;  1904          2028  4                 COMP_LST_SIZE = .RECSTACK[.RSPTR, RS_CLSIZ];
;  1905          2029  4                 REC_DSTPTR = .RECSTACK[.RSPTR, RS_RCDST];
;  1906          2030  4                 VAR_REC_PTR = .RECSTACK[.RSPTR, RS_VRPTR];
;  1907          2031  4                 TYP_RST_SIZE = .RECSTACK[.RSPTR, RS_TRSIZ];
;  1908          2032  3                 END;
;  1909          2033  3
;  1910          2034  3
;  1911          2035  3             ! Handle the Variant-Set Begin record.  This signals the start of a
;  1912          2036  3             ! set of variants of the current record which all have the same tag
;  1913          2037  3             ! variable (though the tag variable need not be materialized).  Here
;  1914          2038  3             ! we find the tag variable and build a Variant RST Entry.  We then
;  1915          2039  3             ! set up a new component list to accumulate the individual variants
;  1916          2040  3             ! of this variant set.
;  1917          2041  3             !
;  1918          2042  3             [DST$K_VARBEG]:
;  1919          2043  4                 BEGIN
;  1920          2044  4                 IF .COMP_LST_PTR EQL 0
;  1921          2045  4                 THEN
;  1922          2046  4                     SIGNAL(DBG$_INCDSTNES, 1, .MODNAMEPTR);
;  1923          2047  4
;  1924          2048  4
;  1925          2049  4                 ! Determine the address of the RST entry of the variant set's
;  1926          2050  4                 ! tag variable if present.  We do this by searching the already
;  1927          2051  4                 ! existing record components for a symbol with the desired DST
;  1928          2052  4                 ! pointer.
;  1929          2053  4                 !
;  1930          2054  4                 VARBEG_DSTPTR = DSTPTR[DST$B_NAME] + 1 + .DSTPTR[DST$B_NAME];
;  1931          2055  4                 TAGDSTPTR = .VARBEG_DSTPTR[ DST$L_VARBEG_TAG_PTR ];
;  1932          2056  4                 TAGRSTPTR = 0;
;  1933          2057  4                 IF .TAGDSTPTR NEQ 0
;  1934          2058  4                 THEN
;  1935          2059  5                     BEGIN
;  1936          2060  5                     TAGDSTPTR = DBG$RST_DST_PTR( .TAGDSTPTR + .DST$BEGIN_ADDR );                    ! MO
;  1937          2061  5                     INCR I FROM 1 TO .COMP_LST_PTR[0] DO
;  1938          2062  6                         BEGIN
;  1939          2063  6                         RSTPTR = .COMP_LST_PTR[.I];
;  1940          2064  6                         IF (.RSTPTR[RST$L_DSTPTR] EQL .TAGDSTPTR) AND
;  1941          2065  7                            (.RSTPTR[RST$B_KIND] EQL RST$K_DATA)
;  1942          2066  6                         THEN
;  1943          2067  7                             BEGIN
;  1944          2068  7                             TAGRSTPTR = .RSTPTR;
;  1945          2069  7                             EXITLOOP;
;  1946          2070  6                             END;
;  1947          2071  6
;  1948          2072  5                         END;
```

```
;  1949     2073  5
;  1950     2074  5                    IF .TAGRSTPTR EQL 0
;  1951     2075  5                    THEN
;  1952     2076  5                        SIGNAL(DBG$_INCDSTNES, 1, .MODNAMEPTR);
;  1953     2077  5
;  1954     2078  4                    END;
;  1955     2079  4
;  1956     2080  4
;  1957     2081  4                ! Build a Variant RST Entry for this variant set.  This entry is
;  1958     2082  4                ! temporary in the sense that it will eventually be copied into
;  1959     2083  4                ! the Type RST Entry for the containing record data type.
;  1960     2084  4                !
;  1961     2085  4                RSTPTR = DBG$GET_MEMORY(RST$K_VARENTSIZ);
;  1962     2086  4                RSTPTR[RST$L_DSTPTR] = .DSTPTR;
;  1963     2087  4                RSTPTR[RST$L_VARTAGPTR] = .TAGRSTPTR;
;  1964     2088  4                RSTPTR[RST$B_KIND] = RST$K_VARIANT;
;  1965     2089  4                COMP_LIST_ADD_ENTRY(COMP_LST_PTR, COMP_LST_SIZE, .RSTPTR);
;  1966     2090  4
;  1967     2091  4
;  1968     2092  4                ! Push the current component list pointer and size on the
;  1969     2093  4                ! record stack and set up a new component list for the
;  1970     2094  4                ! variant set.
;  1971     2095  4                !
;  1972     2096  4                IF .RSPTR EQL MAX_NESTING
;  1973     2097  4                THEN
;  1974     2098  4                    SIGNAL(DBG$_DSTNESDEP, 1, .MODNAMEPTR);
;  1975     2099  4
;  1976     2100  4                RECSTACK[.RSPTR, RS_CLPTR] = .COMP_LST_PTR;
;  1977     2101  4                RECSTACK[.RSPTR, RS_CLSIZ] = .COMP_LST_SIZE;
;  1978     2102  4                RECSTACK[.RSPTR, RS_VRPTR] = .VAR_REC_PTR;
;  1979     2103  4                RSPTR = .RSPTR + 1;
;  1980     2104  4                COMP_LST_PTR = DBG$GET_MEMORY(11);
;  1981     2105  4                COMP_LST_SIZE = 10;
;  1982     2106  4                VAR_REC_PTR = .RSTPTR;
;  1983     2107  4                VARIANT_VALUE_FOUND = FALSE;
;  1984     2108  3                END;
;  1985     2109  3
;  1986     2110  3
;  1987     2111  3            ! Handle the Variant-Set End record.  Close out the component list
;  1988     2112  3            ! of the last variant in the variant set, and then close out the
;  1989     2113  3            ! whole set.
;  1990     2114  3            !
;  1991     2115  3            [DST$K_VAREND]:
;  1992     2116  4                BEGIN
;  1993     2117  4                IF .VAR_REC_PTR EQL 0
;  1994     2118  4                THEN
;  1995     2119  4                    SIGNAL(DBG$_INCDSTNES, 1, .MODNAMEPTR);
;  1996     2120  4
;  1997     2121  4
;  1998     2122  4                    ! Pop the record stack and put the address of the last variant's
;  1999     2123  4                    ! component list on the variant set list.
;  2000     2124  4                    !
;  2001     2125  4                    IF .VARIANT_VALUE_FOUND
;  2002     2126  4                    THEN
;  2003     2127  5                        BEGIN
;  2004     2128  5                        TYP_RST_SIZE = .TYP_RST_SIZE + 1 + .COMP_LST_PTR[0];
;  2005     2129  5                        RSTPTR = .COMP_LST_PTR;
```

```
;  2006          2130  5                 RSPTR = .RSPTR - 1;
;  2007          2131  5                 IF .RSPTR LEQ 0 THEN SIGNAL(DBG$_INCDSTNES, 1, .MODNAMEPTR);
;  2008          2132  5                 COMP_LST_PTR = .RECSTACK[.RSPTR, RS_CLPTR];
;  2009          2133  5                 COMP_LST_SIZE = .RECSTACK[.RSPTR, RS_CLSIZ];
;  2010          2134  5                 COMP_LIST_ADD_ENTRY(COMP_LST_PTR, COMP_LST_SIZE, .RSTPTR);
;  2011          2135  4                 END;
;  2012          2136  4
;  2013          2137  4
;  2014          2138  4             ! Then close out the current variant set by saving the address
;  2015          2139  4             ! of the variant set list in the temporary Variant RST Entry.
;  2016          2140  4             ! Then pop the record stack to continue accumulating record
;  2017          2141  4             ! components for the outer level record definition.
;  2018          2142  4             !
;  2019          2143  4             VARIANT_VALUE_FOUND = TRUE;
;  2020          2144  4             TYP_RST_SIZE = .TYP_RST_SIZE + RST$K_VARENTSIZ + .COMP_LST_PTR[0];
;  2021          2145  4             VAR_REC_PTR[RST$L_SYMCHRPTR] = .COMP_LST_PTR;
;  2022          2146  4             IF .RSPTR LEQ 0 THEN SIGNAL(DBG$_INCDSTNES, 1, .MODNAMEPTR);
;  2023          2147  4             RSPTR = .RSPTR - 1;
;  2024          2148  4             COMP_LST_PTR = .RECSTACK[.RSPTR, RS_CLPTR];
;  2025          2149  4             COMP_LST_SIZE = .RECSTACK[.RSPTR, RS_CLSIZ];
;  2026          2150  4             VAR_REC_PTR = .RECSTACK[.RSPTR, RS_VRPTR];
;  2027          2151  3             END;
;  2028          2152  3
;  2029          2153  3
;  2030          2154  3         ! Handle the Variant-Value record.  This record defines a new vari-
;  2031          2155  3         ! ant of the current record and the corresponding tag value.  Close
;  2032          2156  3         ! out the previous variant (if any) and set things up to accumulate
;  2033          2157  3         ! the components of this new variant.
;  2034          2158  3         !
;  2035          2159  3         [DST$K_VARVAL]:
;  2036          2160  4             BEGIN
;  2037          2161  4             IF .VAR_REC_PTR EQL 0 THEN SIGNAL(DBG$_INCDSTNES,1,.MODNAMEPTR);
;  2038          2162  4
;  2039          2163  4
;  2040          2164  4             ! Unless this is the first variant, pop the record stack and put
;  2041          2165  4             ! the address of the last variant's component list on the vari-
;  2042          2166  4             ! ant set list.
;  2043          2167  4             !
;  2044          2168  4             IF .VARIANT_VALUE_FOUND
;  2045          2169  4             THEN
;  2046          2170  5                 BEGIN
;  2047          2171  5                 TYP_RST_SIZE = .TYP_RST_SIZE + 1 + .COMP_LST_PTR[0];
;  2048          2172  5                 RSTPTR = .COMP_LST_PTR;
;  2049          2173  5                 RSPTR = .RSPTR - 1;
;  2050          2174  5                 IF .RSPTR LEQ 0 THEN SIGNAL(DBG$_INCDSTNES, 1, .MODNAMEPTR);
;  2051          2175  5                 COMP_LST_PTR = .RECSTACK[.RSPTR, RS_CLPTR];
;  2052          2176  5                 COMP_LST_SIZE = .RECSTACK[.RSPTR, RS_CLSIZ];
;  2053          2177  5                 COMP_LIST_ADD_ENTRY(COMP_LST_PTR, COMP_LST_SIZE, .RSTPTR);
;  2054          2178  4                 END;
;  2055          2179  4
;  2056          2180  4
;  2057          2181  4             ! Then set things up to accumulate the components of the new
;  2058          2182  4             ! variant definition.  Push the tag value DST pointer on the
;  2059          2183  4             ! current variant component list, push the record stack, and
;  2060          2184  4             ! set up a component list for the new variant.
;  2061          2185  4             !
;  2062          2186  4             VARIANT_VALUE_FOUND = TRUE;
```

```
;  2063          2187  4                COMP_LIST_ADD_ENTRY(COMP_LST_PTR, COMP_LST_SIZE, .DSTPTR);
;  2064          2188  4                IF .RSPTR EQL MAX_NESTING
;  2065          2189  4                THEN
;  2066          2190  4                    SIGNAL(DBG$_DSTNESDEP, 1, .MODNAMEPTR);
;  2067          2191  4
;  2068          2192  4                RECSTACK[.RSPTR, RS_CLPTR] = .COMP_LST_PTR;
;  2069          2193  4                RECSTACK[.RSPTR, RS_CLSIZ] = .COMP_LST_SIZE;
;  2070          2194  4                RSPTR = .RSPTR + 1;
;  2071          2195  4                COMP_LST_PTR = DBG$GET_MEMORY(11);
;  2072          2196  4                COMP_LST_SIZE = 10;
;  2073          2197  3                END;
;  2074          2198  3
;  2075          2199  3
;  2076          2200  3            ! Handle the Enumeration Type Begin record.  Set up the accumulation
;  2077          2201  3            ! of the enumeration type elements.
;  2078          2202  3            !
;  2079          2203  3            [DST$K_ENUMBEG]:
;  2080          2204  4                BEGIN
;  2081          2205  4
;  2082          2206  4
;  2083          2207  4                ! Set the flag saying that we have to build a Type Reference
;  2084          2208  4                ! Table. (I'm not sure whether this absolutely has to be done
;  2085          2209  4                ! for enumeration types. It does have to be done for records).
;  2086          2210  4                !
;  2087          2211  4                BUILD_TYPE_TABLE = TRUE;
;  2088          2212  4
;  2089          2213  4
;  2090          2214  4                IF .ENUM_LST_PTR NEQ 0
;  2091          2215  4                THEN
;  2092          2216  4                    SIGNAL(DBG$_INCDSTNES, 1, .MODNAMEPTR);
;  2093          2217  4
;  2094          2218  4                ENUM_DSTPTR = .DSTPTR;
;  2095          2219  4                ENUM_BITSIZE = .DSTPTR[DST$B_ENUMBEG_LENG];
;  2096          2220  4                ENUM_LST_PTR = DBG$GET_MEMORY(11);
;  2097          2221  4                ENUM_LST_SIZE = 10;
;  2098          2222  3                END;
;  2099          2223  3
;  2100          2224  3
;  2101          2225  3            ! Handle the Enumeration Type Element record.  Build a Data Item RST
;  2102          2226  3            ! Entry for the element (it constitutes a literal of the enumeration
;  2103          2227  3            ! type).  Also add the element to the type's element list.
;  2104          2228  3            !
;  2105          2229  3            [DST$K_ENUMELT]:
;  2106          2230  4                BEGIN
;  2107          2231  4                IF .ENUM_LST_PTR EQL 0
;  2108          2232  4                THEN
;  2109          2233  4                    SIGNAL(DBG$_INCDSTNES, 1, .MODNAMEPTR);
;  2110          2234  4
;  2111          2235  4                RSTPTR = DBG$GET_MEMORY(RST$K_DATENTSIZ);
;  2112          2236  4                RSTPTR[RST$B_KIND] = RST$K_DATA;
;  2113          2237  4                RSTPTR[RST$L_TYPEPTR] = .ENUM_DSTPTR;
;  2114          2238  4                RSTPTR[RST$V_SET_TYPEPTR] = TRUE;
;  2115          2239  4                NEWENT_FLAG = TRUE;
;  2116          2240  4                COMP_LIST_ADD_ENTRY(ENUM_LST_PTR, ENUM_LST_SIZE, .RSTPTR);
;  2117          2241  4                END;
;  2118          2242  3
;  2119          2243  3
```

```
;  2120          2244  3          ! Handle the Enumeration Type End record.  Build the Data Type RST
;  2121          2245  3          ! Entry for the enumeration type and include the element list as
;  2122          2246  3          ! the type's "component" list.  Then clear ENUM_LST_PTR to indicate
;  2123          2247  3          ! that we are no longer accumulating enumeration type elements.
;  2124          2248  3          !
;  2125          2249  3          [DST$K_ENUMEND]:
;  2126          2250  4              BEGIN
;  2127          2251  4              IF .ENUM_LST_PTR EQL 0
;  2128          2252  4              THEN
;  2129          2253  4                  SIGNAL(DBG$_INCDSTNES, 1, .MODNAMEPTR);
;  2130          2254  4
;  2131          2255  4              RSTPTR = DBG$GET_MEMORY(RST$K_TYPENTSIZ + .ENUM_LST_PTR[0]);
;  2132          2256  4              RSTPTR[RST$B_KIND] = RST$K_TYPE;
;  2133          2257  4              RSTPTR[RST$B_FCODE] = RST$K_TYPE_ENUM;
;  2134          2258  4              RSTPTR[RST$L_BITSIZE] = .ENUM_BITSIZE;
;  2135          2259  4              DSTPTR_TO_SAVE = .ENUM_DSTPTR;
;  2136          2260  4              NEWENT_FLAG = TRUE;
;  2137          2261  4              RSTPTR[RST$L_TYPCOMPCNT] = .ENUM_LST_PTR[0];
;  2138          2262  4              ENUMELT = RSTPTR[RST$A_TYPCOMPLST];
;  2139          2263  4              INCR I FROM 1 TO .ENUM_LST_PTR[0] DO
;  2140          2264  4                  ENUMELT[.I - 1] = .ENUM_LST_PTR[.I];
;  2141          2265  4
;  2142          2266  4              DBG$REL_MEMORY(.ENUM_LST_PTR);
;  2143          2267  4              ENUM_LST_PTR = 0;
;  2144          2268  3              END;
;  2145          2269  3
;  2146          2270  3
;  2147          2271  3          ! Handle all Data Symbol records.  This includes Boolean records and
;  2148          2272  3          ! Separate Type Spec records.  Create a Data Item RST Entry for the
;  2149          2273  3          ! data symbol and make a Static Address Table entry if the data item
;  2150          2274  3          ! is static.
;  2151          2275  3          !
;  2152          2276  3          [DSC$K_DTYPE_LOWEST TO DSC$K_DTYPE_HIGHEST,
;  2153          2277  3           DST$K_BOOL,
;  2154          2278  3           DST$K_SEPTYP]:
;  2155          2279  4              BEGIN
;  2156          2280  4
;  2157          2281  4
;  2158          2282  4              ! Build the Data Item RST Entry.
;  2159          2283  4              !
;  2160          2284  4              RSTPTR = DBG$GET_MEMORY(RST$K_DATENTSIZ);
;  2161          2285  4              RSTPTR[RST$B_KIND] = RST$K_DATA;
;  2162          2286  4              RSTPTR[RST$L_TYPEPTR] = 0;
;  2163          2287  4              NEWENT_FLAG = TRUE;
;  2164          2288  4
;  2165          2289  4
;  2166          2290  4              ! If this is a ZEM data DST, and the name is "TRANSFER$ADDRESS",
;  2167          2291  4              ! and it contains a static address in the VALUE field, then
;  2168          2292  4              ! this is the special DST put out by FORTRAN to tell us where
;  2169          2293  4              ! the transfer address is.
;  2170          2294  4              !
;  2171          2295  4              IF (.DSTPTR[DST$B_TYPE] EQL DSC$K_DTYPE_ZEM) AND
;  2172          2296  4                 (.DSTPTR[DST$B_VFLAGS] EQL DST$K_VALKIND_ADDR) AND
;  2173          2297  4                 CH$EQL(17, DSTPTR[DST$B_NAME],
;  2174          2298  4                        17, UPLIT(%ASCIC 'TRANSFER$ADDRESS'))
;  2175          2299  4              THEN
;  2176          2300  4                  DBG$GL_TRANSFER_ADDRESS = .DSTPTR[DST$L_VALUE];
```

```
;  2177          2301  4
;  2178          2302  4
;  2179          2303  4                    ! If this is a DST$K_SEPTYP record, see if the separate Type
;  2180          2304  4                    ! Spec record ultimately points to a record or enumeration type
;  2181          2305  4                    ! definition.  If so, set up the RST$L_TYPEPTR field in the RST
;  2182          2306  4                    ! entry.  We also set RST$V_MARKBIT if this is an array whose
;  2183          2307  4                    ! cell type is a record type.
;  2184          2308  4                    !
;  2185          2309  4                    IF .DSTPTR[DST$B_TYPE] EQL DST$K_SEPTYP
;  2186          2310  4                    THEN
;  2187          2311  5                        BEGIN
;  2188          2312  5                        GET_TYPEPTR(.DSTPTR, .MODNAMEPTR, TYPEPTR, ARRAY_FLAG);
;  2189          2313  5                        RSTPTR[RST$L_TYPEPTR] = .TYPEPTR;
;  2190          2314  5                        IF .TYPEPTR NEQ 0
;  2191          2315  5                        THEN
;  2192          2316  6                            BEGIN
;  2193          2317  6                            RSTPTR[RST$V_SET_TYPEPTR] = TRUE;
;  2194          2318  6                            BUILD_TYPE_TABLE = TRUE;
;  2195          2319  5                            END;
;  2196          2320  5
;  2197          2321  5                        RSTPTR[RST$V_MARKBIT] = .ARRAY_FLAG;
;  2198          2322  4                        END;
;  2199          2323  4
;  2200          2324  4
;  2201          2325  4                    ! If we are not nested in a record definition and if the data
;  2202          2326  4                    ! item is static, make a Static Address Table entry.
;  2203          2327  4                    !
;  2204          2328  4                    IF .COMP_LST_PTR EQL 0
;  2205          2329  4                    THEN
;  2206          2330  5                        BEGIN
;  2207          2331  5                        STATIC_CHECK(.DSTPTR, ADDRESS, LENGTH, STATIC_FLAG);
;  2208          2332  5                        IF .STATIC_FLAG
;  2209          2333  5                        THEN
;  2210          2334  6                            BEGIN
;  2211          2335  6                            IF .LENGTH NEQ 0 THEN RSTPTR[RST$V_NONZLENGTH] = TRUE;
;  2212          2336  6                            STATIC_ADDRESS(.ADDRESS, .LENGTH, .RSTPTR,
;  2213          2337  6                                           MODRSTPTR[RST$L_SAT_PTR], FALSE);
;  2214          2338  5                            END;
;  2215          2339  5
;  2216          2340  5                        END
;  2217          2341  5
;  2218          2342  5
;  2219          2343  5                    ! If this is a record component, add the data item to the cur-
;  2220          2344  5                    ! rent record's component list.  We do not make a SAT entry for
;  2221          2345  5                    ! a record component (though we may have for the whole record).
;  2222          2346  5                    !
;  2223          2347  4                    ELSE
;  2224          2348  4                        COMP_LIST_ADD_ENTRY(COMP_LST_PTR, COMP_LST_SIZE, .RSTPTR);
;  2225          2349  4
;  2226          2350  3                    END;
;  2227          2351  3
;  2228          2352  3
;  2229          2353  3                ! Handle the BLISS special cases record.  Here we create a Data Item
;  2230          2354  3                ! RST Entry for the BLISS data item.  We also make a Static Address
;  2231          2355  3                ! Table entry if appropriate.
;  2232          2356  3                !
;  2233          2357  3                [DST$K_BLI]:
```

```
;  2234          2358  4                 BEGIN
;  2235          2359  4                 RSTPTR = DBG$GET_MEMORY(RST$K_DATENTSIZ);
;  2236          2360  4                 RSTPTR[RST$B_KIND] = RST$K_DATA;
;  2237          2361  4                 RSTPTR[RST$L_TYPEPTR] = 0;
;  2238          2362  4                 NEWENT_FLAG = TRUE;
;  2239          2363  4                 STATIC_CHECK(.DSTPTR, ADDRESS, LENGTH, STATIC_FLAG);
;  2240          2364  4                 IF .STATIC_FLAG
;  2241          2365  4                 THEN
;  2242          2366  5                     BEGIN
;  2243          2367  5                     IF .LENGTH NEQ 0 THEN RSTPTR[RST$V_NONZLENGTH] = TRUE;
;  2244          2368  5                     STATIC_ADDRESS(.ADDRESS, .LENGTH, .RSTPTR,
;  2245          2369  5                                    MODRSTPTR[RST$L_SAT_PTR], FALSE);
;  2246          2370  4                     END;
;  2247          2371  4
;  2248          2372  3                 END;
;  2249          2373  3
;  2250          2374  3
;  2251          2375  3             ! Handle the BLISS field record.  Just build a Data Item RST Entry.
;  2252          2376  3             !
;  2253          2377  3             [DST$K_BLIFLD]:
;  2254          2378  4                 BEGIN
;  2255          2379  4                 RSTPTR = DBG$GET_MEMORY(RST$K_DATENTSIZ);
;  2256          2380  4                 RSTPTR[RST$B_KIND] = RST$K_DATA;
;  2257          2381  4                 RSTPTR[RST$L_TYPEPTR] = 0;
;  2258          2382  4                 NEWENT_FLAG = TRUE;
;  2259          2383  3                 END;
;  2260          2384  3
;  2261          2385  3             ! Handle the COBOL special cases.  Build a Data Item RST Entry.
;  2262          2386  3             !
;  2263          2387  3             [DST$K_COB_HACK]:
;  2264          2388  4                 BEGIN
;  2265          2389  4                 RSTPTR = DBG$GET_MEMORY(RST$K_DATENTSIZ);
;  2266          2390  4                 RSTPTR[RST$B_KIND] = RST$K_DATA;
;  2267          2391  4                 RSTPTR[RST$L_TYPEPTR] = 0;
;  2268          2392  4                 NEWENT_FLAG = TRUE;
;  2269          2393  4                 END;
;  2270          2394  3
;  2271          2395  3
;  2272          2396  3
;  2273          2397  3             ! Handle the COBOL Global Attribute record.  Here we just set the
;  2274          2398  3             ! COBOL "global" flag.  This flag will then be set in the RST entry
;  2275          2399  3             ! for the next symbol to be defined in the DST.  This attribute is
;  2276          2400  3             ! needed to correctly emulate COBOL's scope rules.
;  2277          2401  3             !
;  2278          2402  3             [DST$K_COBOLGBL]:
;  2279          2403  3                 COBOLGBL_FLAG = TRUE;
;  2280          2404  3
;  2281          2405  3
;  2282          2406  3             ! Handle the Overloaded Symbol DST record.
;  2283          2407  3             !
;  2284          2408  3             [DST$K_OVERLOAD]:
;  2285          2409  4                 BEGIN
;  2286          2410  4                 RSTPTR = DBG$GET_MEMORY(RST$K_OLENTSIZ);
;  2287          2411  4                 RSTPTR[RST$B_KIND] = RST$K_OVERLOAD;
;  2288          2412  4                 NEWENT_FLAG = TRUE;
;  2289          2413  3                 END;
;  2290          2414  3
```

```
;  2291     2415  3
;  2292     2416  3          ! Handle type specs
;  2293     2417  3          !
;  2294     2418  3          [DST$K_TYPSPEC]:
;  2295     2419  4              BEGIN
;  2296     2420  4
;  2297     2421  4              ! Only create an RST entry and hash it in if this type spec
;  2298     2422  4              ! has a name.
;  2299     2423  4              !
;  2300     2424  4              IF .DSTPTR[DST$B_TYPSPEC_NAME] NEQ 0
;  2301     2425  4              THEN
;  2302     2426  5                  BEGIN
;  2303     2427  5
;  2304     2428  5                  ! Get a pointer to the type spec.
;  2305     2429  5                  !
;  2306     2430  5                  TSPTR = .DSTPTR[DST$B_TYPSPEC_NAME] + DSTPTR[DST$B_TYPSPEC_NAME] + 1;
;  2307     2431  5
;  2308     2432  5                  ! Get memory and fill in the fields of the RST type entry
;  2309     2433  5                  !
;  2310     2434  5                  RSTPTR = DBG$GET_MEMORY(RST$K_TYPENTSIZ);
;  2311     2435  5                  RSTPTR[RST$L_DSTPTR] = .DSTPTR;
;  2312     2436  5                  RSTPTR[RST$L_UPSCOPEPTR] = .RST$START_ADDR;
;  2313     2437  5                  RSTPTR[RST$B_KIND] = RST$K_TYPE;
;  2314     2438  5                  RSTPTR[RST$B_FCODE] = DBG$TRANS_TYPE_CODE(.TSPTR[DST$B_TS_KIND]);
;  2315     2439  5
;  2316     2440  5                  ! Note - the code to fill in BITSIZE is commented out.
;  2317     2441  5                  ! This is because the GET_BITSIZE_FROM_TYPESPEC routine
;  2318     2442  5                  ! may, in certain cases, call DBG$STA_VALSPEC to evaluate
;  2319     2443  5                  ! a descriptor to determine the size. If the valspec refers
;  2320     2444  5                  ! to any registers, however, then VALSPEC will signal an
;  2321     2445  5                  ! error, and we cannot allow error signals in this
;  2322     2446  5                  ! initialization code. So we leave BITSIZE zero here.
;  2323     2447  5                  ! (This may affect the "'SIZE" operator in ADA).
;  2324     2448  5                  !
;  2325     2449  5                  ! RSTPTR[RST$L_BITSIZE] = DBG$GET_BITSIZE_FROM_TYPESPEC(.TSPTR);
;  2326     2450  5
;  2327     2451  5                  NEWENT_FLAG = TRUE;
;  2328     2452  4                  END;
;  2329     2453  4
;  2330     2454  3              END;
;  2331     2455  3
;  2332     2456  3
;  2333     2457  3              ! Handle all other cases by ignoring the DST record.
;  2334     2458  3              !
;  2335     2459  3              [INRANGE]:
;  2336     2460  3                  0;
;  2337     2461  3
;  2338     2462  3              TES;                        ! End of CASE over the DST record type
;  2339     2463  3
;  2340     2464  3
;  2341     2465  3          ! If a new RST entry is being created from this DST record, fill in the
;  2342     2466  3          ! remaining fields, link it into the module's symbol chain, and insert
;  2343     2467  3          ! it in the RST Hash Table.
;  2344     2468  3          !
;  2345     2469  3          IF .NEWENT_FLAG
;  2346     2470  3          THEN
;  2347     2471  4              BEGIN
```

```
;  2348          2472  4              RSTPTR[RST$L_DSTPTR] = ,DSTPTR_TO_SAVE;
;  2349          2473  4              RSTPTR[RST$L_UPSCOPEPTR] = .NEST_STACK[.NESTING];
;  2350          2474  4              IF .COBOLGBL_FLAG THEN RSTPTR[RST$V_COBOLGBL] = TRUE;
;  2351          2475  4              COBOLGBL_FLAG = FALSE;
;  2352          2476  4              SYMCHNPTR[RST$L_SYMCHNPTR] = .RSTPTR;
;  2353          2477  4              SYMCHNPTR = .RSTPTR;
;  2354          2478  4              !IF .RSTPTR[RST$B_KIND] EQL RST$K_TYPE
;  2355          2479  4              !THEN
;  2356          2480  4              !    BEGIN
;  2357          2481  4              !    RSTPTR[RST$L_HASH_FLINK] = .RSTPTR;
;  2358          2482  4              !    RSTPTR[RST$L_HASH_BLINK] = .RSTPTR;
;  2359          2483  4              !    END
;  2360          2484  4              !
;  2361          2485  4              !ELSE
;  2362          2486  4              DBG$HASH_INSERT(.RSTPTR);
;  2363          2487  4
;  2364          2488  3              END;
;  2365          2489  3
;  2366          2490  3
;  2367          2491  3          ! If this RST entry defines a new nesting level (either lexical nesting
;  2368          2492  3          ! or data record nesting), we push a new entry on NEST_STACK.
;  2369          2493  3          !
;  2370          2494  3          IF .NEST_FLAG
;  2371          2495  3          THEN
;  2372          2496  4              BEGIN
;  2373          2497  4              IF .NESTING GEQ MAX_NESTING - 1
;  2374          2498  4              THEN
;  2375          2499  4                  SIGNAL(DBG$_DSTNESDEP, 1, .MODNAMEPTR);
;  2376          2500  4
;  2377          2501  4              NESTING = .NESTING + 1;
;  2378          2502  4              NEST_STACK[.NESTING] = .RSTPTR;
;  2379          2503  3              END;
;  2380          2504  3
;  2381          2505  3
;  2382          2506  3          ! If this DST record terminates the current nesting level, pop an entry
;  2383          2507  3          ! off the nesting stack.
;  2384          2508  3          !
;  2385          2509  3          IF .UNNEST_FLAG
;  2386          2510  3          THEN
;  2387          2511  4              BEGIN
;  2388          2512  4              IF .NESTING LEQ 0 THEN SIGNAL(DBG$_INCDSTNES, 1, .MODNAMEPTR);
;  2389          2513  4              NESTING = .NESTING - 1;
;  2390          2514  3              END;
;  2391          2515  3
;  2392          2516  3
;  2393          2517  3          ! Advance the DST pointer to the next sequential DST record and loop.
;  2394          2518  3          !
;  2395          2519  3 !        DSTPTR = DSTPTR[DST$A_NEXT] + .DSTPTR[DST$B_LENGTH];                    ! D015
;  2396          2520  3          DSTPTR = .NEXT_DSTPTR;
;  2397          2521  3
;  2398          2522  2          END;                                ! End of WHILE loop over module's DST
;  2399          2523  2
;  2400          2524  2
;  2401          2525  2      ! If there were any PC-Correlation Table DST records for this module, copy
;  2402          2526  2      ! the table of pointers to those DST records into a separate memory block
;  2403          2527  2      ! and make the RST$L_MODPCTBL field point to that block.  The first long-
;  2404          2528  2      ! word of the block gives the number of DST pointers in the block.  Then
```

```
;  2405          2529  2      ! release the temporary memory block we used for those pointers.
;  2406          2530  2      !
;  2407          2531  2      IF .NUM_PC_TBLS GTR 0
;  2408          2532  2      THEN
;  2409          2533  3          BEGIN
;  2410          2534  3          MODRSTPTR[RST$L_PCTBL_BASE] = .PCTBL_BASE;
;  2411          2535  3          MOD_PC_TBL = DBG$GET_MEMORY(.NUM_PC_TBLS + 1);
;  2412          2536  3          MODRSTPTR[RST$L_MODPCTBL] = .MOD_PC_TBL;
;  2413          2537  3          MOD_PC_TBL[0] = .NUM_PC_TBLS;
;  2414          2538  3          INCR I FROM 1 TO .NUM_PC_TBLS DO
;  2415          2539  3              MOD_PC_TBL[.I] = .PC_CORR_TBL[.I - 1];
;  2416          2540  3
;  2417          2541  2          END;
;  2418          2542  2
;  2419          2543  2      DBG$REL_MEMORY(.PC_CORR_TBL);
;  2420          2544  2
;  2421          2545  2
;  2422          2546  2      ! If there were any Source Line Correlation DST Records for this module,
;  2423          2547  2      ! copy the table of pointers to those DST records into a separate memory
;  2424          2548  2      ! block and make the RST$L_MODPCTBL field point to that block.  The first
;  2425          2549  2      ! longword of the block gives the number of DST pointers in the block.
;  2426          2550  2      ! Then release the temporary memory block we used for those pointers.
;  2427          2551  2      !
;  2428          2552  2      IF .NUM_SRC_DSTS NEQ 0
;  2429          2553  2      THEN
;  2430          2554  3          BEGIN
;  2431          2555  3          MOD_SRC_TBL = DBG$GET_MEMORY(.NUM_SRC_DSTS + 1);
;  2432          2556  3          MODRSTPTR[RST$L_MODSRCTBL] = .MOD_SRC_TBL;
;  2433          2557  3          MOD_SRC_TBL[0] = .NUM_SRC_DSTS;
;  2434          2558  3          INCR I FROM 1 TO .NUM_SRC_DSTS DO
;  2435          2559  3              MOD_SRC_TBL[.I] = .SRC_CORR_TBL[.I - 1];
;  2436          2560  3
;  2437          2561  2          END;
;  2438          2562  2
;  2439          2563  2      DBG$REL_MEMORY(.SRC_CORR_TBL);
;  2440          2564  2
;  2441          2565  2
;  2442          2566  2      ! We must walk the RST to fix any pointers to continued DST records.  We do
;  2443          2567  2      ! it only when there is a continued record at all in the DST.  We can't do
;  2444          2568  2      ! it as we build the RST because there might be pointers in the DST to DST
;  2445          2569  2      ! records that have not been moved at the time the RST entry is built.
;  2446          2570  2      !
;  2447          2571  2      IF .CONTINUATION_DST_SEEN
;  2448          2572  2      THEN
;  2449          2573  3          BEGIN
;  2450          2574  3
;  2451          2575  3
;  2452          2576  3          ! Loop on the RST
;  2453          2577  3          !
;  2454          2578  3          RSTPTR = .MODRSTPTR[RST$L_SYMCHNPTR];
;  2455          2579  3          WHILE .RSTPTR NEQ 0 DO
;  2456          2580  4              BEGIN
;  2457          2581  4
;  2458          2582  4
;  2459          2583  4              ! Transform the DST pointer
;  2460          2584  4              !
;  2461          2585  4              RSTPTR[RST$L_DSTPTR] = DBG$RST_DST_PTR(.RSTPTR[RST$L_DSTPTR]);
```

```
;  2462          2586  4
;  2463          2587  4
;  2464          2588  4              ! RST$L_TYPEPTR may be a DST pointer for data
;  2465          2589  4              !
;  2466          2590  4              IF RSTPTR[RST$B_KIND] EQL RST$K_DATA
;  2467          2591  4              THEN
;  2468          2592  4                  RSTPTR[RST$L_TYPEPTR] = DBG$RST_DST_PTR(.RSTPTR[RST$L_TYPEPTR]);
;  2469          2593  4
;  2470          2594  4
;  2471          2595  4              ! RST$L_DST_TYPE_REC_PTR
;  2472          2596  4              !
;  2473          2597  4              IF RSTPTR[RST$B_KIND] EQL RST$K_TYPE
;  2474          2598  4              THEN
;  2475          2599  4                  RSTPTR[RST$L_DST_TYP_REC_PTR] =
;  2476          2600  4                              DBG$RST_DST_PTR(.RSTPTR[RST$L_DST_TYP_REC_PTR]);
;  2477          2601  4
;  2478          2602  4
;  2479          2603  4              ! Variant RSTs have DST pointers buried in the "Variant Entry"
;  2480          2604  4              !
;  2481          2605  4              IF .RSTPTR[RST$B_KIND] EQL RST$K_TYPE
;  2482          2606  4              THEN
;  2483          2607  5                  BEGIN
;  2484          2608  5
;  2485          2609  5
;  2486          2610  5                  ! Walk the type component list
;  2487          2611  5                  !
;  2488          2612  5                  TYPCOMPLST = RSTPTR[RST$A_TYPCOMPLST];
;  2489          2613  5                  INCR J FROM 0 TO .RSTPTR[RST$L_TYPCOMPCNT] - 1 DO
;  2490          2614  6                      BEGIN
;  2491          2615  6
;  2492          2616  6
;  2493          2617  6                      ! If we find a variant
;  2494          2618  6                      !
;  2495          2619  6                      RST_PTR = .TYPCOMPLST[.J];
;  2496          2620  6                      IF .RST_PTR[RST$B_KIND] EQL RST$K_VARIANT
;  2497          2621  6                      THEN
;  2498          2622  7                          BEGIN
;  2499          2623  7
;  2500          2624  7
;  2501          2625  7                          ! Walk the variant list fixing the DST pointer
;  2502          2626  7                          !
;  2503          2627  7                          VARSETTBL = RST_PTR[RST$A_VARSETTBL];
;  2504          2628  7                          INCR I FROM 0 TO .RST_PTR[RST$L_VARSETCNT] - 1 DO
;  2505          2629  8                              BEGIN
;  2506          2630  8                              VARIANT_ENTRY = .VARSETTBL[.I];
;  2507          2631  8                              VARIANT_ENTRY[RST$L_VAR_DSTPTR] = DBG$RST_DST_PTR(
;  2508          2632  8                                          .VARIANT_ENTRY[RST$L_VAR_DSTPTR]);
;  2509          2633  7                              END;
;  2510          2634  7
;  2511          2635  6                          END;
;  2512          2636  6
;  2513          2637  5                      END;
;  2514          2638  5
;  2515          2639  4                  END;
;  2516          2640  4
;  2517          2641  4
;  2518          2642  4              ! Next RST entry
```

```
;  2519          2643  4              !
;  2520          2644  4              RSTPTR = .RSTPTR[RST$L_SYMCHNPTR];
;  2521          2645  4
;  2522          2646  3              END;
;  2523          2647  3
;  2524          2648  2          END;
;  2525          2649  2
;  2526          2650  2
;  2527          2651  2      ! Go through the module's symbol chain to construct a DST address to RST
;  2528          2652  2      ! address translate table for Data Type RST Entries which refer to records
;  2529          2653  2      ! or enumeration types.
;  2530          2654  2      ! Don't bother to do this unless the BUILD_TYPE_TABLE flag
;  2531          2655  2      ! is set.
;  2532          2656  2      !
;  2533          2657  2      IF .BUILD_TYPE_TABLE
;  2534          2658  2      THEN
;  2535          2659  3          BEGIN
;  2536          2660  3
;  2537          2661  3          TYPE_DST_LST = DBG$GET_MEMORY(11);
;  2538          2662  3          TYPE_DST_LEN = 10;
;  2539          2663  3          TYPE_RST_LST = DBG$GET_MEMORY(11);
;  2540          2664  3          TYPE_RST_LEN = 10;
;  2541          2665  3          RSTPTR = .MODRSTPTR[RST$L_SYMCHNPTR];
;  2542          2666  3          WHILE .RSTPTR NEQ 0 DO
;  2543          2667  4              BEGIN
;  2544          2668  4              IF .RSTPTR[RST$B_KIND] EQL RST$K_TYPE
;  2545          2669  4              THEN
;  2546          2670  5                  BEGIN
;  2547          2671  5                  COMP_LIST_ADD_ENTRY(TYPE_DST_LST, TYPE_DST_LEN,
;  2548          2672  5                                     .RSTPTR[RST$L_DSTPTR]);
;  2549          2673  5                  COMP_LIST_ADD_ENTRY(TYPE_RST_LST, TYPE_RST_LEN, .RSTPTR);
;  2550          2674  4                  END;
;  2551          2675  4
;  2552          2676  4              RSTPTR = .RSTPTR[RST$L_SYMCHNPTR];
;  2553          2677  3              END;
;  2554          2678  3
;  2555          2679  3
;  2556          2680  3          ! Now use this translate table to fill in the RST$L_TYPEPTR field of Data
;  2557          2681  3          ! Item and Type Component RST Entries which reference the Record-Begin or
;  2558          2682  3          ! Enumeration-Type-Begin DST record of their type.
;  2559          2683  3          !
;  2560          2684  3          RSTPTR = .MODRSTPTR[RST$L_SYMCHNPTR];
;  2561          2685  3          WHILE .RSTPTR NEQ 0 DO
;  2562          2686  4              BEGIN
;  2563          2687  4              IF .RSTPTR[RST$V_SET_TYPEPTR]
;  2564          2688  4              THEN
;  2565          2689  5                  BEGIN
;  2566          2690  5                  INCR I FROM 1 TO .TYPE_DST_LST[0] DO
;  2567          2691  6                      BEGIN
;  2568          2692  6                      IF .TYPE_DST_LST[.I] EQL .RSTPTR[RST$L_TYPEPTR]
;  2569          2693  6                      THEN
;  2570          2694  7                          BEGIN
;  2571          2695  7                          RSTPTR[RST$L_TYPEPTR] = .TYPE_RST_LST[.I];
;  2572          2696  7                          RSTPTR[RST$V_SET_TYPEPTR] = FALSE;
;  2573          2697  7                          EXITLOOP;
;  2574          2698  6                          END;
;  2575          2699  6
```

```
;  2576          2700  5                 END;
;  2577          2701  5
;  2578          2702  5                 IF .RSTPTR[RST$V_SET_TYPEPTR]
;  2579          2703  5                 THEN
;  2580          2704  5                     SIGNAL(DBG$_INCDSTNES, 1, .MODNAMEPTR);
;  2581          2705  5
;  2582          2706  4                 END;
;  2583          2707  4
;  2584          2708  4             RSTPTR = .RSTPTR[RST$L_SYMCHNPTR];
;  2585          2709  3             END;
;  2586          2710  3
;  2587          2711  3         ! Go through the symbol chain to find all Data RST Entries and all Type
;  2588          2712  3         ! Component RST Entries which point to Type RST Entries for Record data
;  2589          2713  3         ! types.  For each such RST entry, increment the RST$W_TYPREFCNT field
;  2590          2714  3         ! in the corresponding Type RST Entry.  At the end of the loop, we will
;  2591          2715  3         ! thus know how many RST entries reference each Record Type RST Entry.
;  2592          2716  3         ! This is the information we need to allocate space for the Type Refer-
;  2593          2717  3         ! ence Table for each Record Type RST Entry.
;  2594          2718  3         !
;  2595          2719  3         RSTPTR = .MODRSTPTR[RST$L_SYMCHNPTR];
;  2596          2720  3         WHILE .RSTPTR NEQ 0 DO
;  2597          2721  4             BEGIN
;  2598          2722  4             IF (.RSTPTR[RST$B_KIND] EQL RST$K_DATA) OR
;  2599          2723  5                (.RSTPTR[RST$B_KIND] EQL RST$K_TYPCOMP)
;  2600          2724  4             THEN
;  2601          2725  5                 BEGIN
;  2602          2726  5                 IF .RSTPTR[RST$L_TYPEPTR] NEQ 0
;  2603          2727  5                 THEN
;  2604          2728  6                     BEGIN
;  2605          2729  6                     TYPEPTR = .RSTPTR[RST$L_TYPEPTR];
;  2606          2730  6                     IF .TYPEPTR[RST$B_KIND] NEQ RST$K_TYPE
;  2607          2731  6                     THEN
;  2608          2732  6                         $DBG_ERROR('RSTCNTRL\RST_BUILD 20');
;  2609          2733  6
;  2610          2734  6                     IF .TYPEPTR[RST$B_FCODE] EQL RST$K_TYPE_RECORD
;  2611          2735  6                     THEN
;  2612          2736  6                         TYPEPTR[RST$W_TYPREFCNT] = .TYPEPTR[RST$W_TYPREFCNT] + 1;
;  2613          2737  6
;  2614          2738  5                     END;
;  2615          2739  5
;  2616          2740  4                 END;
;  2617          2741  4
;  2618          2742  4             RSTPTR = .RSTPTR[RST$L_SYMCHNPTR];
;  2619          2743  3             END;
;  2620          2744  3
;  2621          2745  3
;  2622          2746  3         ! Now loop through all Type RST Entries, and allocate a "Type Reference
;  2623          2747  3         ! Table" for each such entry with a non-zero reference count.  The type
;  2624          2748  3         ! reference table for each entry will be used to store pointers to all
;  2625          2749  3         ! RST entries which reference this Type RST Entry (i.e., to all symbols
;  2626          2750  3         ! of this data type).
;  2627          2751  3         !
;  2628          2752  3         INCR I FROM 1 TO .TYPE_RST_LST[0] DO
;  2629          2753  4             BEGIN
;  2630          2754  4             TYPEPTR = .TYPE_RST_LST[.I];
;  2631          2755  4             IF .TYPEPTR[RST$W_TYPREFCNT] NEQ 0
;  2632          2756  4             THEN
```

```
;  2633          2757  5                 BEGIN
;  2634          2758  5                 TYPEPTR[RST$L_TYPREFTBL] = DBG$GET_MEMORY(.TYPEPTR[RST$W_TYPREFCNT]);
;  2635          2759  5                 TYPEPTR[RST$W_TYPREFCNT] = 0;
;  2636          2760  4                 END;
;  2637          2761  4
;  2638          2762  3             END;
;  2639          2763  3
;  2640          2764  3
;  2641          2765  3         ! To complete the "Type Reference Tables", go through the module's symbol
;  2642          2766  3         ! chain once more.  For each Data or Type Component RST Entry with a type
;  2643          2767  3         ! pointer to a Record Type RST Entry, add that entry's RST pointer to the
;  2644          2768  3         ! Type RST Entry's Type Reference Table.  This data structure is later
;  2645          2769  3         ! used by DBG$STA_GETSYMBOL to find all records which could have a given
;  2646          2770  3         ! symbol as a record component.
;  2647          2771  3         !
;  2648          2772  3         RSTPTR = .MODRSTPTR[RST$L_SYMCHNPTR];
;  2649          2773  3         WHILE .RSTPTR NEQ 0 DO
;  2650          2774  4             BEGIN
;  2651          2775  4             IF (.RSTPTR[RST$B_KIND] EQL RST$K_DATA) OR
;  2652          2776  5                (.RSTPTR[RST$B_KIND] EQL RST$K_TYPCOMP)
;  2653          2777  4             THEN
;  2654          2778  5                 BEGIN
;  2655          2779  5                 IF .RSTPTR[RST$L_TYPEPTR] NEQ 0
;  2656          2780  5                 THEN
;  2657          2781  6                     BEGIN
;  2658          2782  6
;  2659          2783  6
;  2660          2784  6                     ! This RST entry has a type pointer.  Add a pointer to the
;  2661          2785  6                     ! corresponding Type RST Entry's Type Reference Table which
;  2662          2786  6                     ! points back to this current RST entry.
;  2663          2787  6                     !
;  2664          2788  6                     TYPEPTR = .RSTPTR[RST$L_TYPEPTR];
;  2665          2789  6                     IF .TYPEPTR[RST$B_FCODE] EQL RST$K_TYPE_RECORD
;  2666          2790  6                     THEN
;  2667          2791  7                         BEGIN
;  2668          2792  7                         TYPREFTBL = .TYPEPTR[RST$L_TYPREFTBL];
;  2669          2793  7                         TYPREFTBL[.TYPEPTR[RST$W_TYPREFCNT]] = .RSTPTR;
;  2670          2794  7                         TYPEPTR[RST$W_TYPREFCNT] = .TYPEPTR[RST$W_TYPREFCNT] + 1;
;  2671          2795  6                         END;
;  2672          2796  6
;  2673          2797  6
;  2674          2798  6                     ! If RST$V_MARKBIT is set, this RST entry is for an array of
;  2675          2799  6                     ! records and RST$L_TYPEPTR points to the cell type of the
;  2676          2800  6                     ! array.  This pointer is no longer needed and in fact should
;  2677          2801  6                     ! no longer be there.  We thus clear it and the mark bit.
;  2678          2802  6                     !
;  2679          2803  6                     IF .RSTPTR[RST$V_MARKBIT]
;  2680          2804  6                     THEN
;  2681          2805  7                         BEGIN
;  2682          2806  7                         RSTPTR[RST$L_TYPEPTR] = 0;
;  2683          2807  7                         RSTPTR[RST$V_MARKBIT] = FALSE;
;  2684          2808  6                         END;
;  2685          2809  6
;  2686          2810  5                 END;
;  2687          2811  5
;  2688          2812  4             END;
;  2689          2813  4
```

```
;  2690          2814  4              RSTPTR = .RSTPTR[RST$L_SYMCHNPTR];
;  2691          2815  3              END;
;  2692          2816  3
;  2693          2817  3
;  2694          2818  3          ! Now release the DST and RST address tables for types to the memory pool.
;  2695          2819  3          !
;  2696          2820  3          DBG$REL_MEMORY(.TYPE_DST_LST);
;  2697          2821  3          DBG$REL_MEMORY(.TYPE_RST_LST);
;  2698          2822  2          END;
;  2699          2823  2
;  2700          2824  2
;  2701          2825  2      ! We are then all done, and can return to the caller.
;  2702          2826  2      !
;  2703          2827  2      MODRSTPTR[RST$V_MOD_IN_RST] = TRUE;
;  2704          2828  2      RETURN;
;  2705          2829  1      END;
;  INFO#250                L1:2125
;  Referenced LOCAL symbol VARIANT_VALUE_FOUND is probably not initialized
;  INFO#250                L1:2168
;  Referenced LOCAL symbol VARIANT_VALUE_FOUND is probably not initialized
;  INFO#250                L1:2237
;  Referenced LOCAL symbol ENUM_DSTPTR is probably not initialized
;  INFO#250                L1:2258
;  Referenced LOCAL symbol ENUM_BITSIZE is probably not initialized
;  INFO#250                L1:2259
;  Referenced LOCAL symbol ENUM_DSTPTR is probably not initialized


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

42  5F  54  53  52  5C  4C  52  54  4E  43  54  53  52  15  00087 P.AAJ:  .ASCII  <21>\RSTCNTRL\<92>\RST_BUILD 10\                ;
                        30  31  20  44  4C  49  55  00096                                                                    ;
                                                    0009D       .BLKB   3                                                    ;
45  52  44  44  41  24  52  45  46  53  4E  41  52  54  10  000A0 P.AAK:  .ASCII  <16>\TRANSFER$ADDRESS\<0><0><0>             ;
                                    00  00  00  53  53  000AF                                                                ;
42  5F  54  53  52  5C  4C  52  54  4E  43  54  53  52  15  000B4 F.AAL:  .ASCII  <21>\RSTCNTRL\<92>\RST_BUILD 20\            ;
                        30  32  20  44  4C  49  55  000C3                                                                    ;


                                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                    0FFC 00000          .ENTRY  DBG$RST_BUILD, Save R2,R3,R4,R5,R6,R7,R8,-       ; 1275
                                                                                R9,R10,R11                                       ;
                                5E      F9C8    CE  9E 00002            MOVAB   -1592(SP), SP                                    ;
                                                7E  D4 00007            CLRL    CONTINUATION_DST_SEEN                            ; 1294
                                6D      0FF6    CF  DE 00009            MOVAL   150$, (FP)                                       ;
                00000000'       EF      04      AC  D0 0000E            MOVL    MODRSTPTR, MOD_BEING_BUILT                       ; 1437
            50          04      AC              14  C1 00016            ADDL3   #20, MODRSTPTR, R0                               ; 1443
                                01              60  91 0001B            CMPB    (R0), #1                                         ;
                                                15  13 0001E            BEQL    1$                                               ;
                                    00000000'   EF  9F 00020            PUSHAB  P.AAJ                                            ; 1445
                                                01  DD 00026            PUSHL   #1                                               ;
                                    00028362    8F  DD 00028            PUSHL   #164706                                          ;
                00000000G       00              03  FB 0002E            CALLS   #3, LIB$SIGNAL                                   ;
            50          04      AC              28  C1 00035 1$:         ADDL3   #40, MODRSTPTR, R0                               ; 1447
```

```
                        50   AE                60  9E 0003A          MOVAB   (R0), 80(SP)
                    01  50   BE                01  E1 0003E          BBC     #1, @80(SP), 2$
                                                04  00043          RET
                        50   BE                02  88 00044 2$:      BISB2   #2, @80(SP)                 1448
                             04   AC           DD  00048          PUSHL   MODRSTPTR                   1454
                        0000V CF                01  FB 0004B          CALLS   #1, DBG$RST_MOST_RECENT
                    29  50   BE                04  E1 00050          BBC     #4, @80(SP), 4$         1459
                             04   AC           DD  00055          PUSHL   MODRSTPTR                   1463
                    52  04   AC                30  C1 00058          ADDL3   #48, MODRSTPTR, R2      1462
                                                62  DD 0005D          PUSHL   (R2)
                    53  04   AC                34  C1 0005F          ADDL3   #52, MODRSTPTR, R3
                             7E                63  9A 00064          MOVZBL  (R3), -(SP)
                        FA9E CF                03  FB 00067          CALLS   #3, DBG$GST_INIT
                             01                50  E9 0006C          BLBC    R0, 3$
                                                04  0006F          RET
                                 00028FAB      8F  DD 00070 3$:      PUSHL   #167851                     1465
                00000000G        00            01  FB 00076          CALLS   #1, LIB$SIGNAL
                                                04  0007D          RET                                 1461
                    50   04   AC                0C  C1 0007E 4$:      ADDL3   #12, MODRSTPTR, R0      1476
                        008C CE                 60  D0 00083          MOVL    (R0), DSTPTR
                             52        008C     CE  D0 00088          MOVL    DSTPTR, R2              1477
                             BC   8F        01   A2  91 0008D          CMPB    1(R2), #188
                                                0D  13 00092          BEQL    5$
                                 0002832A       8F  DD 00094          PUSHL   #164650
                00000000G        00            01  FB 0009A          CALLS   #1, LIB$SIGNAL
    50  BE              08                  08        03   A2  F0 000A1 5$:      INSV    3(R2), #8, #8, @80(SP)  1478
                             57                 07   A2  9E 000A8          MOVAB   7(R2), MODNAMEPTR       1479
                                                52  DD 000AC          PUSHL   R2                          1480
                        0000V CF                01  FB 000AE          CALLS   #1, DBG$RST_NEXT_DST
                        008C CE                 50  D0 000B3          MOVL    R0, DSTPTR
        05      50  BE              08          08  ED 000B8          CMPZV   #8, #8, @80(SP), #5     1488
                                                04  12 000BE          BNEQ    6$
                             50   BE            20  88 000C0          BISB2   #32, @80(SP)            1490
                    50       04   AC            18  C1 000C4 6$:      ADDL3   #24, MODRSTPTR, R0      1496
                             1C   AE            60  9E 000C9          MOVAB   (R0), 28(SP)
                                       1C       BE  D4 000CD          CLRL    @28(SP)
                             24   AE       04   AC  D0 000D0          MOVL    MODRSTPTR, SYMCHNPTR    1497
                                           68   AE  D4 000D5          CLRL    COBOLGBL_FLAG           1498
                                                59  D4 000D8          CLRL    NUM_PC_TBLS             1506
                             48   AE       64   8F  9A 000DA          MOVZBL  #100, PC_CORR_SIZE      1507
                                           48   AE  DD 000DF          PUSHL   PC_CORR_SIZE            1508
                        0000V CF                01  FB 000E2          CALLS   #1, GET_MEM
                             34   AE            50  D0 000E7          MOVL    R0, PC_CORR_TBL
                                                58  D4 000EB          CLRL    NUM_SRC_DSTS            1509
                             44   AE       64   8F  9A 000ED          MOVZBL  #100, SRC_CORR_SIZE     1510
                                           44   AE  DD 000F2          PUSHL   SRC_CORR_SIZE           1511
                        0000V CF                01  FB 000F5          CALLS   #1, GET_MEM
                             30   AE            50  D0 000FA          MOVL    R0, SRC_CORR_TBL
                             5C   AE  7FFFFFFF  8F  D0 000FE          MOVL    #2147483647, PCTBL_BASE 1512
                        FF38 CD         04      AC  D0 00106          MOVL    MODRSTPTR, NEST_STACK   1519
                                      FE70      CD  D4 0010C          CLRL    NEST_BREAK_STACK        1525
                                                56  D4 00110          CLRL    RSPTR                   1531
                                      00A0      CE  7C 00112          CLRQ    COMP_LST_SIZE           1533
                                        0C      AE  7C 00116          CLRQ    NESTING_BREAK           1524
                                        18      AE  D4 00119          CLRL    VAR_REC_PTR             1535
                                        04      AE  7C 0011C          CLRQ    NESTING                 1518
                                      0094      CE  D4 0011F          CLRL    ENUM_LST_PTR            1537
```

```
                                    6C   AE   D4 00123                CLRL    BUILD_TYPE_TABLE                        ; 1545
                 0D 00000000G       00        01 E1 00126 7$:         BBC     #1, DBG$GV_CONTROL+1, 8$                ; 1554
                            000280E8 8F        DD 0012E                PUSHL   #164072                                 ;
                 00000000G          00        01 FB 00134             CALLS   #1, LIB$SIGNAL                          ;
                                    14   AE   D4 0013B 8$:           CLRL    NEWENT_FLAG                             ; 1565
                                    60   AE   D4 0013E               CLRL    NEST_FLAG                               ; 1566
                                    54   AE   D4 00141               CLRL    UNNEST_FLAG                             ; 1567
                                 74   AE   BC   8F 9A 00144          MOVZBL  #188, DST_TYPE                          ; 1568
                                 50   008C CE   D0 00149             MOVL    DSTPTR, R0                              ; 1569
                 00000000'       EF        50 D1 0014E             CMPL    R0, DST$END_ADDR                        ;
                                           05 1E 00155             BGEQU   9$                                      ;
                                 74   AE   01   A0 9A 00157         MOVZBL  1(R0), DST_TYPE                         ; 1571
                                      0088 CE   9F 0015C 9$:        PUSHAB  NEXT_DSTPTR                             ; 1577
                                      0090 CE   9F 00160            PUSHAB  DSTPTR                                  ;
                              0000V   CF        02 FB 00164         CALLS   #2, MERGE_CONTINUATION_DST               ;
                                      03        50 E9 00169         BLBC    R0, 10$                                 ;
                                      6E        01 D0 0016C         MOVL    #1, CONTINUATION_DST_SEEN               ; 1579
                                      5B   008C CE   D0 0016F 10$:  MOVL    DSTPTR, R11                             ; 1581
                                      2C   AE   5B   D0 00174       MOVL    R11, DSTPTR_TO_SAVE                     ;
                          000000FF    8F        00   74 AE CF 00178 CASEL   DST_TYPE, #0, #255                      ; 1586
                0A0C            0A0C            0A0C      0AA9      00181 11$:    .WORD   92$-11$,-                       ;
                0A0C            0A0C            0A0C      0A0C      00189         .WORD   86$-11$,-                       ;
                0A0C            0A0C            0A0C      0A0C      00191         .WORD   86$-11$,-                       ;
                0A0C            0A0C            0A0C      0A0C      00199         .WORD   86$-11$,-                       ;
                0A0C            0A0C            0A0C      0A0C      001A1         .WORD   86$-11$,-                       ;
                0A0C            0A0C            0A0C      0A0C      001A9         .WORD   86$-11$,-                       ;
                0A0C            0A0C            0A0C      0A0C      001B1         .WORD   86$-11$,-                       ;
                0A0C            0A0C            0A0C      0A0C      001B9         .WORD   86$-11$,-                       ;
                0A0C            0A0C            0A0C      0A0C      001C1         .WORD   86$-11$,-                       ;
                0B67            0B67            0A0C      0A0C      001C9         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      001D1         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      001D9         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      C867      001E1         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      001E9         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      001F1         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      001F9         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      00201         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      00209         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      00211         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      00219         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      00221         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      00229         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      00231         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      00239         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      00241         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      00249         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      00251         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      00259         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      00261         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      00269         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      00271         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      00279         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      00281         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      00289         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      00291         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      00299         .WORD   86$-11$,-                       ;
                0B67            0B67            0B67      0B67      002A1         .WORD   86$-11$,-                       ;
```

```
0B67    0B67    0B67    0B67    002A9           86$-11$,-
02EE    0B0D    022A    0B67    002B1           103$-11$,-
0B67    0A0C    0866    0467    002B9           103$-11$,-
0A0C    0454    0B67    0B67    002C1           103$-11$,-
06A9    09A4    0928    0963    002C9           103$-11$,-
0513    0B67    0B13    07A3    002D1           103$-11$,-
0B25    0B67    0B67    05F3    002D9           103$-11$,-
0B67    0AF8    03FD    03D9    002E1           103$-11$,-
0AF8    02B0    04DE    0B67    002E9           103$-11$,-
048F    04A9    02B0    0238    002F1           103$-11$,-
03AB    032D    020C    0200    002F9           103$-11$,-
0B67    0B67    0B67    0B67    00301           103$-11$,-
0B67    0B67    0B67    0B67    00309           103$-11$,-
0B67    0B67    0B67    0B67    00311           103$-11$,-
0B67    0B67    0B67    0B67    00319           103$-11$,-
0B67    0B67    0B67    0B67    00321           103$-11$,-
0B67    0B67    0B67    0B67    00329           103$-11$,-
0B67    0B67    0B67    0B67    00331           103$-11$,-
0B67    0B67    0B67    0B67    00339           103$-11$,-
0B67    0B67    0B67    0B67    00341           103$-11$,-
0B67    0B67    0B67    0B67    00349           103$-11$,-
0B67    0B67    0B67    0B67    00351           103$-11$,-
0B67    0B67    0B67    0B67    00359           103$-11$,-
0B67    0B67    0B67    0B67    00361           103$-11$,-
0B67    0B67    0B67    0B67    00369           103$-11$,-
0B67    0B67    0B67    0B67    00371           103$-11$,-
0B67    0B67    0B67    0B67    00379           103$-11$,-
                                                103$-11$,-
                                                103$-11$,-
                                                
```

```
103$-11$,-
103$-11$,-
103$-11$,-
```

```
103$-11$,-
103$-11$,-
17$-11$,-
98$-11$,-
25$-11$,-
44$-11$,-
72$-11$,-
86$-11$,-
103$-11$,-
103$-11$,-
103$-11$,-
42$-11$,-
86$-11$,-
80$-11$,-
77$-11$,-
82$-11$,-
60$-11$,-
67$-11$,-
100$-11$,-
103$-11$,-
52$-11$,-
57$-11$,-
103$-11$,-
103$-11$,-
101$-11$,-
36$-11$,-
38$-11$,-
97$-11$,-
103$-11$,-
103$-11$,-
50$-11$,-
23$-11$,-
97$-11$,-
18$-11$,-
23$-11$,-
48$-11$,-
47$-11$,-
12$-11$,-
13$-11$,-
28$-11$,-
34$-11$,-
103$-11$,-
103$-11$,-
103$-11$,-
103$-11$,-
103$-11$,-
103$-11$,-
103$-11$,-
103$-11$,-
103$-11$,-
103$-11$,-
103$-11$,-
103$-11$,-
103$-11$,-
103$-11$,-
103$-11$,-
```

```
                                                103$-11$,-
                                                103$-11$
                57 DD 00381 12$:        PUSHL   MODNAMEPTR                      1595
                01 DD 00383             PUSHL   #1
0002864B        8F DD 00385             PUSHL   #165451
                14 11 0038B             BRB     15$
          04    AE D5 0038D 13$:        TSTL    NESTING                         1605
                05 12 00390             BNEQ    14$
          0C    AE D5 00392             TSTL    NESTING_BREAK
                11 13 00395             BEQL    16$
                57 DD 00397 14$:        PUSHL   MODNAMEPTR                      1607
```

```
                                                01 DD 00399          PUSHL   #1                                                      ;
                                      0002896A  8F DD 00039B         PUSHL   #166250                                                 ;
                          00000000G   00        03 FB 003A1 15$:     CALLS   #3, LIB$SIGNAL                                          ;
                                              09BE 31 003A8 16$:     BRW     11$                                                     ; 1604
                05        50    BE            08 ED 003AB 17$      CMPZV   #8, #8, @80(SP), #5                                     ; 1621
                                                7C 12 003B1          BNEQ    22$                                                     ;
                          50    BE            20 8A 003B3          BICB2   #32, @80(SP)                                            ; 1623
                                                76 11 003B7          BRB     22$                                                     ; 1586
                        FF00    8F      50    BE B3 003B9 18$:     BITW    @80(SP), #65280                                         ; 1633
                                                6E 12 003BF          BNEQ    22$                                                     ;
                                                5B DD 003C1          PUSHL   R11                                                     ; 1642
                        F4E5    CF            01 FB 003C3          CALLS   #1, DBG$GET_DST_NAME                                    ;
                        0080    CE            50 D0 003C8          MOVL    R0, NAMEPTR                                             ;
                                      0080    CE DD 003CD          PUSHL   NAMEPTR                                                 ; 1643
                        FB4A    CF            01 FB 003D1          CALLS   #1, DBG$HASH_FIND_SETUP                                 ;
                                      0080    CE DD 003D6 19$:     PUSHL   NAMEPTR                                                 ; 1646
                        FAC4    CF            01 FB 003DA          CALLS   #1, DBG$HASH_FIND                                       ;
                                5A            50 D0 003DF          MOVL    R0, RSTPTR                                              ;
                                              3E 12 003E2          BNEQ    21$                                                     ; 1653
                                              07 DD 003E4          PUSHL   #7                                                      ; 1656
                          00000000G   00        01 FB 003E6          CALLS   #1, DBG$GET_MEMORY                                      ;
                                5A            50 D0 003ED          MOVL    R0, RSTPTR                                              ;
                              14    AA            04 90 003F0          MOVB    #4, 20(RSTPTR)                                          ; 1657
                              18    AA      03    AB D0 003F4          MOVL    3(R11), 24(RSTPTR)                                      ; 1658
                              14    AE            01 D0 003F9          MOVL    #1, NEWENT_FLAG                                         ; 1659
                                    50      07    AB 9A 003FD          MOVZBL  7(R11), R0                                              ; 1661
                              7C    AE      08 A04B 9E 00401           MOVAB   8(R0)[R11], PPTR                                        ;
                            00AC    CE      7C    BE D0 00407          MOVL    @PPTR, LENGTH                                           ; 1662
                              15    AA            02 88 0040D          BISB2   #2, 21(RSTPTR)                                          ; 1663
                                              7E D4 00411          CLRL    -(SP)                                                   ; 1665
                                        20    AE DD 00413          PUSHL   32(SP)                                                  ;
                                              5A DD 00416          PUSHL   RSTPTR                                                  ;
                                      00B8    CE DD 00418          PUSHL   LENGTH                                                  ;
                                        18    AA DD 0041C 20$:     PUSHL   24(RSTPTR)                                              ;
                                            0850 31 0041F          BRW     96$                                                     ;
                                    50        04    AE D0 00422 21$:     MOVL    NESTING, R0                                             ; 1675
                                FF38  CD40    10    AA D1 00426          CMPL    16(RSTPTR), NEST_STACK[R0]                              ;
                                              A7 12 0042B          BNEQ    19$                                                     ;
                                              7A 11 0042F 22$:     BRB     27$                                                     ; 1677
                              48    AE              59 D1 00431 23$:     CMPL    NUM_PC_TBLS, PC_CORR_SIZE                               ; 1694
                                              2F 19 00435          BLSS    24$                                                     ;
                              28    AE        34    AE D0 00437          MOVL    PC_CORR_TBL, MOD_PC_TBL                                 ; 1697
                              48    AE  00000064    8F C0 0043C          ADDL2   #100, PC_CORR_SIZE                                      ; 1698
                                        48    AE DD 00444          PUSHL   PC_CORR_SIZE                                            ; 1699
                          00000000G   00        01 FB 00447          CALLS   #1, DBG$GET_MEMORY                                      ;
                              34    AE              50 D0 0044E          MOVL    R0, PC_CORR_TBL                                         ;
                    50                            02 78 00452          ASHL    #2, NUM_PC_TBLS, R0                                     ; 1700
          34    BE        28    BE                50 28 00456          MOVC3   R0, @MOD_PC_TBL, @PC_CORR_TBL                           ;
                                        28    AE DD 0045C          PUSHL   MOD_PC_TBL                                              ; 1701
                          00000000G   00        01 FB 0045F          CALLS   #1, DBG$REL_MEMORY                                      ;
                              34  BE49              5B D0 00466 24$:     MOVL    R11, @PC_CORR_TBL[NUM_PC_TBLS]                          ; 1704
                                              59 D6 0046B          INCL    NUM_PC_TBLS                                             ; 1705
                                              3C 11 0046D          BRB     27$                                                     ; 1586
                              44    AE              58 D1 0046F 25$:     CMPL    NUM_SRC_DSTS, SRC_CORR_SIZE                             ; 1716
                                              2F 19 00473          BLSS    26$                                                     ;
                              64    AE        30    AE D0 00475          MOVL    SRC_CORR_TBL, MOD_SRC_TBL                               ; 1719
                              44    AE  00000064    8F C0 0047A          ADDL2   #100, SRC_CORR_SIZE                                     ; 1720
```

```
                                    44    AE   DD 00482          PUSHL   SRC_CORR_SIZE                                  ; 1721
                        00000000G   00        01 FB 00485          CALLS   #1, DBG$GET_MEMORY                            ;
                              30    AE        50 D0 0048C          MOVL    R0, SRC_CORR_TBL                              ;
                        50          58        02 78 00490          ASHL    #2, NUM_SRC_DSTS, R0                          ; 1722
                30 BE         64    BE        50 28 00494          MOVC3   R0, @MOD_SRC_TBL, @SRC_CORR_TBL               ;
                                    64    AE   DD 0049A          PUSHL   MOD_SRC_TBL                                   ; 1723
                        00000000G   00        01 FB 0049D          CALLS   #1, DBG$REL_MEMORY                            ;
                              30 BE48         5B D0 004A4 26$:     MOVL    R11, @SRC_CORR_TBL[NUM_SRC_DSTS]              ; 1726
                                              58 D6 004A9          INCL    NUM_SRC_DSTS                                  ; 1727
                                          083A 31 004AB 27$:       BRW     103$                                          ; 1586
                                              0B DD 004AE 28$:     PUSHL   #11                                           ; 1737
                        00000000G   00        01 FB 004B0          CALLS   #1, DBG$GET_MEMORY                            ;
                                    5A        50 D0 004B7          MOVL    R0, RSTPTR                                    ;
                                    14    AA  02 90 004BA          MOVB    #2, 20(RSTPTR)                                ; 1738
                                    52    18  AA 9E 004BE          MOVAB   24(RSTPTR), R2                                ; 1739
                                    62    03  AB D0 004C2          MOVL    3(R11), (R2)                                  ;
                                    1C    AA  62 D0 004C6          MOVL    (R2), 28(RSTPTR)                              ; 1740
                                    32    0C  AE D1 004CA          CMPL    NESTING_BREAK, #50                            ; 1741
                                              11 19 004CE          BLSS    29$                                           ;
                                              57 DD 004D0          PUSHL   MODNAMEPTR                                    ; 1743
                                              01 DD 004D2          PUSHL   #1                                            ;
                                    00028962  8F DD 004D4          PUSHL   #166242                                       ;
                        00000000G   00        03 FB 004DA          CALLS   #3, LIB$SIGNAL                                ;
                              FF00  8F        50 BE B3 004E1 29$:  BITW    @80(SP), #65280                               ; 1747
                                              03 13 004E7          BEQL    30$                                           ;
                                              0C AE D6 004E9       INCL    NESTING_BREAK                                 ; 1749
                                    50        0C AE D0 004EC 30$:  MOVL    NESTING_BREAK, R0                             ; 1751
                                 FE70 CD40        5A D0 004F0      MOVL    RSTPTR, NEST_BREAK_STACK[R0]                  ;
                                              02 AB 95 004F6       TSTB    2(R11)                                        ; 1752
                                              06 18 004F9          BGEQ    31$                                           ;
                                    28    AA  62 D0 004FB          MOVL    (R2), 40(RSTPTR)                              ; 1754
                                              05 11 004FF          BRB     32$                                           ;
                                28  AA        62       02 C1 00501 31$:  ADDL3   #2, (R2), 40(RSTPTR)                  ; 1757
                                    14    AE  01 D0 00506 32$:  MOVL    #1, NEWENT_FLAG                               ; 1759
                                    03    AB        5C AE D1 0050A       CMPL    PCTBL_BASE, 3(R11)                    ; 1760
                                              05 15 0050F          BLEQ    33$                                           ;
                                    5C    AE        03 AB D0 00511       MOVL    3(R11), PCTBL_BASE                    ; 1762
                                 FF00  8F        50 BE B3 00516 33$:  BITW    @80(SP), #65280                               ; 1764
                                              5A 12 0051C          BNEQ    37$                                           ;
                                              7E D4 0051E          CLRL    -(SP)                                         ; 1767
                                          20  AE DD 00520          PUSHL   32(SP)                                        ;
                                              5A DD 00523          PUSHL   RSTPTR                                        ;
                                              7E D4 00525          CLRL    -(SP)                                         ;
                                              62 DD 00527          PUSHL   (R2)                                          ;
                                          0746 31 00529          BRW     96$                                           ;
                                          0C  AE D7 0052C 34$:     DECL    NESTING_BREAK                                 ; 1780
                                    50    04  AE D0 0052F          MOVL    NESTING, R0                                   ; 1781
                                    5A    FF38 CD40 D0 00533       MOVL    NEST_STACK[R0], RSTPTR                        ;
                                    02    14  AA 91 00539          CMPB    20(RSTPTR), #2                                ; 1782
                                              11 13 0053D          BEQL    35$                                           ;
                                              57 DD 0053F          PUSHL   MODNAMEPTR                                    ; 1784
                                              01 DD 00541          PUSHL   #1                                            ;
                                    000289CA  8F DD 00543          PUSHL   #166250                                       ;
                        00000000G   00        03 FB 00549          CALLS   #3, LIB$SIGNAL                                ;
                              00AC  CE        03 AB D0 00550 35$:  MOVL    3(R11), LENGTH                                ; 1786
                                              4F 15 00556          BLEQ    40$                                           ; 1787
                                              52 11 00558          BRB     41$                                           ; 1788
```

```
                                 08  DD 0055A 36$:    PUSHL   #8                                      ; 1801
          00000000G   00         01  FB 0055C         CALLS   #1, DBG$GET_MEMORY                     ;
                      5A         50  D0 00563         MOVL    R0, RSTPTR                             ;
                14    AA         03  90 00566         MOVB    #3, 20(RSTPTR)                         ; 1802
                18    AA      03 AB  D0 0056A         MOVL    3(R11), 24(RSTPTR)                     ; 1803
                1C    AA      18 AA  D0 0056F         MOVL    24(RSTPTR), 28(RSTPTR)                 ; 1804
                14    AE         01  D0 00574         MOVL    #1, NEWENT_FLAG                        ; 1805
                60    AE         01  D0 00578 37$:    MOVL    #1, NEST_FLAG                          ; 1806
                                 68  11 0057C         BRB     43$                                    ; 1586
                      50      04 AE  D0 0057E 38$:    MOVL    NESTING, R0                            ; 1816
                      5A    FF38 CD40 D0 00582         MOVL    NEST_STACK[R0], RSTPTR                 ;
                      03      14 AA  91 00588         CMPB    20(RSTPTR), #3                         ; 1817
                                 11  13 0058C         BEQL    39$                                    ;
                                 57  DD 0058E         PUSHL   MODNAMEPTR                             ; 1819
                                 01  DD 00590         PUSHL   #1                                     ;
                         0002896A 8F  DD 00592         PUSHL   #166250                                ;
          00000000G   00         03  FB 00598         CALLS   #3, LIB$SIGNAL                         ;
             00AC     CE      03 AB  D0 0059F 39$:    MOVL    3(R11), LENGTH                         ; 1821
                                 05  14 005A5         BGTR    41$                                    ; 1822
             00AC     CE         01  D0 005A7 40$:    MOVL    #1, LENGTH                             ;
          50    18    AA   00AC CE  C1 005AC 41$:    ADDL3   LENGTH, 24(RSTPTR), R0                 ; 1823
                1C    AA      FF A0  9E 005B3         MOVAB   -1(R0), 28(RSTPTR)                     ;
                15    AA         02  88 005B8         BISB2   #2, 21(RSTPTR)                         ; 1824
                                 7E  D4 005BC         CLRL    -(SP)                                  ; 1826
                               20 AE  DD 005BE         PUSHL   32(SP)                                 ;
                                 5A  DD 005C1         PUSHL   RSTPTR                                 ;
                            00B8 CE  DD 005C3         PUSHL   LENGTH                                 ;
                               18 AA  DD 005C7         PUSHL   24(RSTPTR)                             ;
             0000V  CF              05  FB 005CA         CALLS   #5, STATIC_ADDRESS                     ;
                54    AE         01  D0 005CF         MOVL    #1, UNNEST_FLAG                        ; 1827
                                 38  11 005D3         BRB     46$                                    ; 1586
                      50      0C AE  D0 005D5 42$:    MOVL    NESTING_BREAK, R0                      ; 1839
                      5A    FE70 CD40 D0 005D9         MOVL    NEST_BREAK_STACK[R0], RSTPTR           ;
                                 2C  13 005DF         BEQL    46$                                    ; 1840
                28    AA      02 AB  D0 005E1         MOVL    2(R11), 40(RSTPTR)                     ; 1842
                                 25  11 005E6 43$:    BRB     46$                                    ; 1586
                      50      04 AE  D0 005E8 44$:    MOVL    NESTING, R0                            ; 1853
                      5A    FF38 CD40 D0 005EC         MOVL    NEST_STACK[R0], RSTPTR                 ;
                      02      14 AA  91 005F2         CMPB    20(RSTPTR), #2                         ; 1854
                                 11  13 005F6         BEQL    45$                                    ;
                                 57  DD 005F8         PUSHL   MODNAMEPTR                             ; 1856
                                 01  DD 005FA         PUSHL   #1                                     ;
                         0002896A 8F  DD 005FC         PUSHL   #166250                                ;
          00000000G   00         03  FB 00602         CALLS   #3, LIB$SIGNAL                         ;
                24    AA         5B  D0 00609 45$:    MOVL    R11, 36(RSTPTR)                        ; 1858
                                 06D8 31 0060D 46$:    BRW     103$                                   ; 1586
                                 5B  DD 00610 47$:    PUSHL   R11                                    ; 1867
             0000V  CF              01  FB 00612         CALLS   #1, BUILD_LABEL_RST                    ;
                      5A         50  D0 00617         MOVL    R0, RSTPTR                             ;
                2C    AE      0C AA  D0 0061A         MOVL    12(RSTPTR), DSTPTR_TO_SAVE             ; 1868
                14    AA         04  90 0061F         MOVB    #4, 20(RSTPTR)                         ; 1869
                18    AA      03 AB  D0 00623         MOVL    3(R11), 24(RSTPTR)                     ; 1870
                                 5A  11 00628         BRB     51$                                    ; 1871
                      01      02 AB  91 0062A 48$:    CMPB    2(R11), #1                             ; 1883
                                 03  13 0062E         BEQL    49$                                    ;
                                 0646 31 00630         BRW     97$                                    ;
                                 5B  DD 00633 49$:    PUSHL   R11                                    ; 1886
```

```
                0000V  CF              01  FB 00635          CALLS   #1, BUILD_LABEL_RST                    ;
                       5A              50  D0 0063A          MOVL    R0, RSTPTR                            ;
                  2C   AE        0C    AA  D0 0063D          MOVL    12(RSTPTR), DSTPTR_TO_SAVE            ; 1887
                  14   AA              04  90 00642          MOVB    #4, 20(RSTPTR)                        ; 1888
                  18   AA        03    AB  D0 00646          MOVL    3(R11), 24(RSTPTR)                    ; 1889
                                        7E  D4 0064B          CLRL    -(SP)                                 ; 1891
                                  20    AE  DD 0064D          PUSHL   32(SP)                                ;
                                        5A  DD 00650          PUSHL   RSTPTR                                ;
                                        7E  D4 00652          CLRL    -(SP)                                 ;
                                  18    AA  DD 00654          PUSHL   24(RSTPTR)                            ;
                0000V  CF               05  FB 00657          CALLS   #5, STATIC_ADDRESS                    ;
                                      0685  31 0065C          BRW     10$                                   ; 1883
                                        0B  DD 0065F  50$:    PUSHL   #11                                   ; 1910
          00000000G    00               01  FB 00661          CALLS   #1, DBG$GET_MEMORY                    ;
                       5A               50  D0 00668          MOVL    R0, RSTPTR                            ;
                  14   AA               08  90 0066B          MOVB    #8, 20(RSTPTR)                        ; 1911
                  18   AA        03     AB  D0 0066F          MOVL    3(R11), 24(RSTPTR)                    ; 1912
                       50        0C     AE  D0 00674          MOVL    NESTING_BREAK, R0                     ; 1913
                  FE70  CD40             5A  D0 00678          MOVL    RSTPTR, NEST_BREAK_STACK[R0]          ;
          28   AA     18   AA            02  C1 0067E          ADDL3   #2, 24(RSTPTR), 40(RSTPTR)            ; 1914
                  14   AE               01  D0 00684  51$:    MOVL    #1, NEWENT_FLAG                       ; 1915
                                        7E  D4 00688          CLRL    -(SP)                                 ; 1917
                                  20    AE  DD 0068A          PUSHL   32(SP)                                ;
                                        5A  DD 0068D          PUSHL   RSTPTR                                ;
                                        7E  D4 0068F          CLRL    -(SP)                                 ;
                                      FD88  31 00691          BRW     20$                                   ;
                  6C   AE               01  D0 00694  52$:    MOVL    #1, BUILD_TYPE_TABLE                  ; 1934
                  80   8F        02     AB  91 00698          CMPB    2(R11), #T28                          ; 1940
                                        69  13 0069D          BEQL    55$                                   ;
                                        07  DD 0069F          PUSHL   #7                                    ; 1943
          00000000G    00               01  FB 006A1          CALLS   #1, DBG$GET_MEMORY                    ;
                       5A               50  D0 006A8          MOVL    R0, RSTPTR                            ;
                  14   AA               06  90 006AB          MOVB    #6, 20(RSTPTR)                        ; 1944
                  18   AA               5B  D0 006AF          MOVL    R11, 24(RSTPTR)                       ; 1945
                  15   AA               08  88 006B3          BISB2   #8, 21(RSTPTR)                        ; 1946
                  14   AE               01  D0 006B7          MOVL    #1, NEWENT_FLAG                       ; 1947
                                  00A4  CE  D5 006BB          TSTL    COMP_LST_PTR                          ; 1948
                                        38  12 006BF          BNEQ    54$                                   ;
                                  00A8  CE  9F 006C1          PUSHAB  STATIC_FLAG                           ; 1951
                                  00B0  CE  9F 006C5          PUSHAB  LENGTH                                ;
                                  00B8  CE  9F 006C9          PUSHAB  ADDRESS                               ;
                                        5B  DD 006CD          PUSHL   R11                                   ;
                0000V  CF               04  FB 006CF          CALLS   #4, STATIC_CHECK                      ;
                       2F         00A8  CE  E9 006D4          BLBC    STATIC_FLAG, 55$                      ; 1952
                                  00AC  CE  D5 006D9          TSTL    LENGTH                                ; 1955
                                        04  13 006DD          BEQL    53$                                   ;
                  15   AA               02  88 006DF          BISB2   #2, 21(RSTPTR)                        ;
                                        7E  D4 006E3  53$:    CLRL    -(SP)                                 ; 1957
                                    20  AE  DD 006E5          PUSHL   32(SP)                                ;
                                        5A  DD 006E8          PUSHL   RSTPTR                                ;
                                  00B8  CE  DD 006EA          PUSHL   LENGTH                                ;
                                  00C0  CE  DD 006EE          PUSHL   ADDRESS                               ;
                0000V  CF               05  FB 006F2          CALLS   #5, STATIC_ADDRESS                    ;
                                        0F  11 006F7          BRB     55$                                   ; 1948
                                        5A  DD 006F9  54$:    PUSHL   RSTPTR                                ; 1963
                                  00A4  CE  9F 006FB          PUSHAB  COMP_LST_SIZE                         ;
                                  00AC  CE  9F 006FF          PUSHAB  COMP_LST_PTR                          ;
```

```
                0000V  CF              03  FB 00703          CALLS   #3, COMP_LIST_ADD_ENTRY
                    32                 56  D1 00708 55$:      CMPL    RSPTR, #50                          ; 1972
                                       11  19 0070B          BLSS    56$
                                       57  DD 0070D          PUSHL   MODNAMEPTR                          ; 1974
                                       01  DD 0070F          PUSHL   #1
                          00028962     8F  DD 00711          PUSHL   #166242
                00000000G   00         03  FB 00717          CALLS   #3, LIB$SIGNAL
            50                 56      14  C5 0071E 56$:      MULL3   #20, RSPTR, R0                      ; 1976
                                  00C4 CE40 9F 00722          PUSHAB  RECSTACK[R0]
                               9E 00A8  CE  D0 00727          MOVL    COMP_LST_PTR, @(SP)+
                                  00C8 CE40 9F 0072C          PUSHAB  RECSTACK+4[R0]                      ; 1977
                               9E 00A4  CE  D0 00731          MOVL    COMP_LST_SIZE, @(SP)+
                                  00CC CE40 9F 00736          PUSHAB  RECSTACK+8[R0]                      ; 1978
                               9E   14  AE  D0 0073B          MOVL    REC_DSTPTR, @(SP)+
                                  00D0 CE40 9F 0073F          PUSHAB  RECSTACK+12[R0]                     ; 1979
                               9E   1C  AE  D0 00744          MOVL    VAR_REC_PTR, @(SP)+
                                  00D4 CE40 9F 00748          PUSHAB  RECSTACK+16[R0]                     ; 1980
                               9E   0C  AE  D0 0074D          MOVL    TYP_RST_SIZE, @(SP)+
                                       56  D6 00751          INCL    RSPTR                               ; 1981
                                       0B  DD 00753          PUSHL   #11                                 ; 1982
                00000000G   00         01  FB 00755          CALLS   #1, DBG$GET_MEMORY
                    00A4    CE         50  D0 0075C          MOVL    R0, COMP_LST_PTR
                    00A0    CE         0A  D0 00761          MOVL    #10, COMP_LST_SIZE                  ; 1983
                      10    AE         5B  D0 00766          MOVL    R11, REC_DSTPTR                     ; 1984
                                  18   AE  D4 0076A          CLRL    VAR_REC_PTR                         ; 1985
                            08    AE   0B  D0 0076D          MOVL    #11, TYP_RST_SIZE                   ; 1986
                                  0574 31  00771          BRW     103$                                ; 1586
                                  18   AE  D5 00774 57$:      TSTL    VAR_REC_PTR                         ; 1996
                                       11  13 00777          BEQL    58$
                                       57  DD 00779          PUSHL   MODNAMEPTR                          ; 1998
                                       01  DD 0077B          PUSHL   #1
                          0002896A     8F  DD 0077D          PUSHL   #166250
                00000000G   00         03  FB 00783          CALLS   #3, LIB$SIGNAL
                      08    AE   00A4   DE  C0 0078A 58$:      ADDL2   @COMP_LST_PTR, TYP_RST_SIZE         ; 2003
                                    08   AE  DD 00790          PUSHL   TYP_RST_SIZE                        ; 2004
                00000000G   00         01  FB 00793          CALLS   #1, DBG$GET_MEMORY
                            5A         50  D0 0079A          MOVL    R0, RSTPTR
                      14    AA         07  90 0079D          MOVB    #7, 20(RSTPTR)                      ; 2005
                      18    AA         07  90 007A1          MOVB    #7, 24(RSTPTR)                      ; 2006
            51        10    AE         07  C1 007A5          ADDL3   #7, REC_DSTPTR, R1                  ; 2007
                            50         61  9A 007AA          MOVZBL  (R1), R0
                            51   10    AE  D0 007AD          MOVL    REC_DSTPTR, R1
                      78    AE   08  A041  9E 007B1          MOVAB   8(R0)[R1], RECBEGTRLR
                      20    AA   78    BE  D0 007B7          MOVL    @RECBEGTRLR, 32(RSTPTR)             ; 2008
                      2C    AE   10    AE  D0 007BC          MOVL    REC_DSTPTR, DSTPTR_TO_SAVE          ; 2009
                      14    AE         01  D0 007C1          MOVL    #1, NEWENT_FLAG                     ; 2010
                00000000'   EF         5A  D0 007C5          MOVL    RSTPTR, TYPRSTPTR                   ; 2017
                00000000'   EF   28    AA  9E 007CC          MOVAB   40(R10), TYPRSTLOC                  ; 2018
                                  00A4 CE  DD 007D4          PUSHL   COMP_LST_PTR                        ; 2019
                            0000V  CF      01  FB 007D8          CALLS   #1, COPY_COMP_LIST
                                           56  D5 007DD          TSTL    RSPTR                           ; 2025
                                           11  14 007DF          BGTR    59$
                                           57  DD 007E1          PUSHL   MODNAMEPTR
                                           01  DD 007E3          PUSHL   #1
                                  0002896A     8F  DD 007E5          PUSHL   #166250
                                00000000G   00     03  FB 007EB          CALLS   #3, LIB$SIGNAL
                                               56  D7 007F2 59$:      DECL    RSPTR                       ; 2026
```

```
50              56              14   C5 007F4          MULL3   #20, RSPTR, R0                          ; 2027
                        00C4 CE40    9F 007F8          PUSHAB  RECSTACK[R0]                            ;
        00A4    CE                   9E D0 007FD       MOVL    @(SP)+, COMP_LST_PTR                    ;
                        00C8 CE40    9F 00802          PUSHAB  RECSTACK+4[R0]                          ; 2028
        00A0    CE                   9E D0 00807       MOVL    @(SP)+, COMP_LST_SIZE                   ;
                        00CC CE40    9F 0080C          PUSHAB  RECSTACK+8[R0]                          ; 2029
          10    AE                   9E D0 00811       MOVL    @(SP)+, REC_DSTPTR                      ;
                        00D0 CE40    9F 00815          PUSHAB  RECSTACK+12[R0]                         ; 2030
          18    AE                   9E D0 0081A       MOVL    @(SP)+, VAR_REC_PTR                     ;
                        00D4 CE40    9F 0081E          PUSHAB  RECSTACK+16[R0]                         ; 2031
          08    AE                   9E D0 00823       MOVL    @(SP)+, TYP_RST_SIZE                    ;
                             04BE    31 00827          BRW     103$                                    ; 1586
                52      00A4   CE    D0 0082A 60$:     MOVL    COMP_LST_PTR, R2                        ; 2044
                               11    12 0082F          BNEQ    61$                                     ;
                               57    DD 00831          PUSHL   MODNAMEPTR                              ; 2046
                               01    DD 00833          PUSHL   #1                                      ;
                        0002896A     8F DD 00835       PUSHL   #166250                                 ;
    00000000G   00                   03 FB 0083B       CALLS   #3, LIB$SIGNAL                          ;
                50          07   AB  9A 00842 61$:     MOVZBL  7(R11), R0                              ; 2054
          3C    AE          08 A04B  9E 00846          MOVAB   8(R0)[R11], VARBEG_DSTPTR               ;
50        3C    AE                   04 C1 0084C       ADDL3   #4, VARBEG_DSTPTR, R0                   ; 2055
          20    AE                   60 D0 00851       MOVL    (R0), TAGDSTPTR                         ;
                             58    AE D4 00855          CLRL    TAGRSTPTR                               ; 2056
                             20    AE D5 00858          TSTL    TAGDSTPTR                               ; 2057
                               49    13 0085B          BEQL    65$                                     ;
                50          20    AE D0 0085D          MOVL    TAGDSTPTR, R0                           ; 2060
                        00000000'FF40    9F 00861       PUSHAB  @DST$BEGIN_ADDR[R0]                     ;
        0000V   CF                   01 FB 00868       CALLS   #1, DBG$RST_DST_PTR                     ;
          20    AE                   50 D0 0086D       MOVL    R0, TAGDSTPTR                           ;
                               50    D4 00871          CLRL    I                                       ; 2064
                               17    11 00873          BRB     63$                                     ;
                5A           6240    D0 00875 62$:     MOVL    (R2)[I], RSTPTR                         ; 2063
          20    AE          0C   AA  D1 00879          CMPL    12(RSTPTR), TAGDSTPTR                   ; 2064
                               0C    12 0087E          BNEQ    63$                                     ;
                06          14   AA  91 00880          CMPB    20(RSTPTR), #6                          ; 2065
                               06    12 00884          BNEQ    63$                                     ;
          58    AE                   5A D0 00886       MOVL    RSTPTR, TAGRSTPTR                       ; 2068
                               04    11 0088A          BRB     64$                                     ; 2067
E5              50                   62 F3 0088C 63$:  AOBLEQ  (R2), I, 62$                           ; 2061
                             58    AE D5 00890 64$:     TSTL    TAGRSTPTR                               ; 2074
                               11    12 00893          BNEQ    65$                                     ;
                               57    DD 00895          PUSHL   MODNAMEPTR                              ; 2076
                               01    DD 00897          PUSHL   #1                                      ;
                        0002896A     8F DD 00899       PUSHL   #166250                                 ;
    00000000G   00                   03 FB 0089F       CALLS   #3, LIB$SIGNAL                          ;
                               06    DD 008A6 65$:     PUSHL   #6                                      ; 2085
    00000000G   00                   01 FB 008A8       CALLS   #1, DBG$GET_MEMORY                      ;
                5A                   50 D0 008AF       MOVL    R0, RSTPTR                              ;
          0C    AA                   5B D0 008B2       MOVL    R11, 12(RSTPTR)                         ; 2086
          10    AA          58   AE  D0 008B6          MOVL    TAGRSTPTR, 16(RSTPTR)                   ; 2087
          14    AA                   0B 90 008BB       MOVB    #11, 20(RSTPTR)                         ; 2088
                               5A    DD 008BF          PUSHL   RSTPTR                                  ; 2089
                        00A4   CE    9F 008C1          PUSHAB  COMP_LST_SIZE                           ;
                        00AC   CE    9F 008C5          PUSHAB  COMP_LST_PTR                            ;
        0000V   CF                   03 FB 008C9       CALLS   #3, COMP_LIST_ADD_ENTRY                 ;
                32                   56 D1 008CE       CMPL    RSPTR, #50                              ; 2096
                               11    12 008D1          BNEQ    66$                                     ;
```

```
                          57  DD 008D3          PUSHL   MODNAMEPTR                              ; 2098
                          01  DD 008D5          PUSHL   #1                                      ;
                00028962  8F  DD 008D7          PUSHL   #166242                                 ;
    00000000G   00        03  FB 008DD          CALLS   #3, LIB$SIGNAL                          ;
50              56        14  C5 008E4 66$:     MULL3   #20, RSPTR, R0                          ; 2100
                      00C4 CE40 9F 008E8        PUSHAB  RECSTACK[R0]                            ;
                9E    00A8  CE  D0 008ED        MOVL    COMP_LST_PTR, @(SP)+                    ;
                      00C8 CE40 9F 008F2        PUSHAB  RECSTACK+4[R0]                          ; 2101
                9E    00A4  CE  D0 008F7        MOVL    COMP_LST_SIZE, @(SP)+                   ;
                      00D0 CE40 9F 008FC        PUSHAB  RECSTACK+12[R0]                         ; 2102
                9E      1C  AE  D0 00901        MOVL    VAR_REC_PTR, @(SP)+                     ;
                          56  D6 00905          INCL    RSPTR                                   ; 2103
                          0B  DD 00907          PUSHL   #11                                     ; 2104
    00000000G   00        01  FB 00909          CALLS   #1, DBG$GET_MEMORY                      ;
      00A4      CE          50  D0 00910        MOVL    R0, COMP_LST_PTR                        ;
      00A0      CE          0A  D0 00915        MOVL    #10, COMP_LST_SIZE                      ; 2105
        18      AE          5A  D0 0091A        MOVL    RSTPTR, VAR_REC_PTR                     ; 2106
                      38    AE  D4 0091E        CLRL    VARIANT_VALUE_FOUND                     ; 2107
                    03C4    31  00921           BRW     103$                                    ; 1586
                      18    AE  D5 00924 67$:   TSTL    VAR_REC_PTR                             ; 2117
                          11  12 00927          BNEQ    68$                                     ;
                          57  DD 00929          PUSHL   MODNAMEPTR                              ; 2119
                          01  DD 0092B          PUSHL   #1                                      ;
                0002896A  8F  DD 0092D          PUSHL   #166250                                 ;
    00000000G   00        03  FB 00933          CALLS   #3, LIB$SIGNAL                          ;
                50        38  AE  E9 0093A 68$:   BLBC    VARIANT_VALUE_FOUND, 70$            ; 2125
                51      00A4  CE  D0 0093E        MOVL    COMP_LST_PTR, R1                        ; 2128
                50            61  D0 00943      MOVL    (R1), R0                                ;
                52        08  AE  D0 00946        MOVL    TYP_RST_SIZE, R2                        ;
          08    AE        01  A042 9E 0094A     MOVAB   1(R0)[R2], TYP_RST_SIZE                 ;
                5A            51  D0 00950      MOVL    R1, RSTPTR                              ; 2129
                11            56  F5 00953      SOBGTR  RSPTR, 69$                              ; 2130
                              57  DD 00956      PUSHL   MODNAMEPTR                              ; 2131
                              01  DD 00958      PUSHL   #1                                      ;
                    0002896A  8F  DD 0095A      PUSHL   #166250                                 ;
        00000000G   00        03  FB 00960      CALLS   #3, LIB$SIGNAL                          ;
50              56            14  C5 00967 69$:   MULL3   #20, RSPTR, R0                      ; 2132
                      00C4 CE40 9F 0096B        PUSHAB  RECSTACK[R0]                            ;
          00A4      CE          9E  D0 00970    MOVL    @(SP)+, COMP_LST_PTR                    ;
                      00C8 CE40 9F 00975        PUSHAB  RECSTACK+4[R0]                          ; 2133
          00A0      CE          9E  D0 0097A    MOVL    @(SP)+, COMP_LST_SIZE                   ;
                              5A  DD 0097F      PUSHL   RSTPTR                                  ; 2134
                      00A4    CE  9F 00981      PUSHAB  COMP_LST_SIZE                           ;
                      00AC    CE  9F 00985      PUSHAB  COMP_LST_PTR                            ;
          0000V     CF            03  FB 00989    CALLS   #3, COMP_LIST_ADD_ENTRY             ;
            38      AE            01  D0 0098E 70$:   MOVL    #1, VARIANT_VALUE_FOUND         ; 2143
                51      00A4  CE  D0 00992        MOVL    COMP_LST_PTR, R1                        ; 2144
                50            61  D0 00997      MOVL    (R1), R0                                ;
                52        08  AE  D0 0099A        MOVL    TYP_RST_SIZE, R2                        ;
            08  AE        06  A042 9E 0099E     MOVAB   6(R0)[R2], TYP_RST_SIZE                 ;
50          18  AE            08  C1 009A4      ADDL3   #8, VAR_REC_PTR, R0                     ; 2145
                60            51  D0 009A9      MOVL    R1, (R0)                                ;
                              56  D5 009AC      TSTL    RSPTR                                   ; 2146
                              11  14 009AE      BGTR    71$                                     ;
                              57  DD 009B0      PUSHL   MODNAMEPTR                              ;
                              01  DD 009B2      PUSHL   #1                                      ;
                    0002896A  8F  DD 009B4      PUSHL   #166250                                 ;
```

```
                00000000G   00          03   FB 009BA         CALLS   #3, LIB$SIGNAL
                                        56   D7 009C1 71$:    DECL    RSPTR                                   ;2147
                50              56      14   C5 009C3         MULL3   #20, RSPTR, R0                         ;2148
                                    00C4 CE40   9F 009C7         PUSHAB  RECSTACK[R0]
                        00A4    CE          9E   D0 009CC         MOVL    @(SP)+, COMP_LST_PTR
                                    00C8 CE40   9F 009D1         PUSHAB  RECSTACK+4[R0]                      ;2149
                        00A0    CE          9E   D0 009D6         MOVL    @(SP)+, COMP_LST_SIZE
                                    00D0 CE40   9F 009DB         PUSHAB  RECSTACK+12[R0]                     ;2150
                          18    AE          9E   D0 009E0         MOVL    @(SP)+, VAR_REC_PTR
                                        0301    31 009E4         BRW     103$                                ;1586
                                  18    AE    D5 009E7 72$:    TSTL    VAR_REC_PTR                             ;2161
                                        11    12 009EA         BNEQ    73$
                                        57    DD 009EC         PUSHL   MODNAMEPTR
                                        01    DD 009EE         PUSHL   #1
                                0002896A      8F   DD 009F0         PUSHL   #166250
                00000000G   00              03   FB 009F6         CALLS   #3, LIB$SIGNAL
                                    38    AE    E9 009FD 73$:    BLBC    VARIANT_VALUE_FOUND, 75$            ;2169
                                    00A4  CE    D0 00A01         MOVL    COMP_LST_PTR, R1                    ;2171
                                    61    D0 00A06         MOVL    (R1), R0
                                    08    AE    D0 00A09         MOVL    TYP_RST_SIZE, R2
                          08    AE      01 A042   9E 00A0D         MOVAB   1(R0)[R2], TYP_RST_SIZE
                                    51    D0 00A13         MOVL    R1, RSTPTR                          ;2172
                                    56    F5 00A16         SOBGTR  RSPTR, 74$                          ;2173
                                    57    DD 00A19         PUSHL   MODNAMEPTR                          ;2174
                                    01    DD 00A1B         PUSHL   #1
                                0002896A      8F   DD 00A1D         PUSHL   #166250
                00000000G   00              03   FB 00A23         CALLS   #3, LIB$SIGNAL
                50              56          14   C5 00A2A 74$:    MULL3   #20, RSPTR, R0                      ;2175
                                    00C4 CE40   9F 00A2E         PUSHAB  RECSTACK[R0]
                        00A4    CE          9E   D0 00A33         MOVL    @(SP)+, COMP_LST_PTR
                                    00C8 CE40   9F 00A38         PUSHAB  RECSTACK+4[R0]                      ;2176
                        00A0    CE          9E   D0 00A3D         MOVL    @(SP)+, COMP_LST_SIZE
                                    5A    DD 00A42         PUSHL   RSTPTR                              ;2177
                                    00A4  CE    9F 00A44         PUSHAB  COMP_LST_SIZE
                                    00AC  CE    9F 00A48         PUSHAB  COMP_LST_PTR
                        0000V   CF          03   FB 00A4C         CALLS   #3, COMP_LIST_ADD_ENTRY
                          38    AE          01   D0 00A51 75$:    MOVL    #1, VARIANT_VALUE_FOUND             ;2186
                                    5B    DD 00A55         PUSHL   R11                                 ;2187
                                    00A4  CE    9F 00A57         PUSHAB  COMP_LST_SIZE
                                    00AC  CE    9F 00A5B         PUSHAB  COMP_LST_PTR
                        0000V   CF          03   FB 00A5F         CALLS   #3, COMP_LIST_ADD_ENTRY
                                32          56   D1 00A64         CMPL    RSPTR, #50                          ;2188
                                        11    12 00A67         BNEQ    76$
                                        57    DD 00A69         PUSHL   MODNAMEPTR                          ;2190
                                        01    DD 00A6B         PUSHL   #1
                                00028962      8F   DD 00A6D         PUSHL   #166242
                00000000G   00              03   FB 00A73         CALLS   #3, LIB$SIGNAL
                50              56          14   C5 00A7A 76$:    MULL3   #20, RSPTR, R0                      ;2192
                                    00C4 CE40   9F 00A7E         PUSHAB  RECSTACK[R0]
                                9E  00A8  CE    D0 00A83         MOVL    COMP_LST_PTR, @(SP)+
                                    00C8 CE40   9F 00A88         PUSHAB  RECSTACK+4[R0]                      ;2193
                                9E  00A4  CE    D0 00A8D         MOVL    COMP_LST_SIZE, @(SP)+
                                    56    D6 00A92         INCL    RSPTR                               ;2194
                                    0B    DD 00A94         PUSHL   #11                                 ;2195
                00000000G   00              01   FB 00A96         CALLS   #1, DBG$GET_MEMORY
                        00A4    CE          50   D0 00A9D         MOVL    R0, COMP_LST_PTR
                        00A0    CE          0A   D0 00AA2         MOVL    #10, COMP_LST_SIZE                  ;2196
```

```
                                38  11 00AA7                BRB     79$                                     ; 1586
                             6C AE      01  D0 00AA9 77$:    MOVL    #1, BUILD_TYPE_TABLE                    ; 2211
                          0094  CE      D5  00AAD            TSTL    ENUM_LST_PTR                            ; 2214
                                        11  13 00AB1        BEQL    78$
                                        57  DD 00AB3        PUSHL   MODNAMEPTR                              ; 2216
                                        01  DD 00AB5        PUSHL   #1
                          0002896A      8F  DD 00AB7        PUSHL   #166250
                00000000G   00          03  FB 00ABD        CALLS   #3, LIB$SIGNAL
                      70    AE          5B  D0 00AC4 78$:    MOVL    R11, ENUM_DSTPTR                        ; 2218
                    0084    CE    02    AB  9A 00AC8        MOVZBL  2(R11), ENUM_BITSIZE                    ; 2219
                                        0B  DD 00ACE        PUSHL   #11                                     ; 2220
                00000000G   00          01  FB 00AD0        CALLS   #1, DBG$GET_MEMORY
                    0094    CE          50  D0 00AD7        MOVL    R0, ENUM_LST_PTR
                    0090    CE          0A  D0 00ADC        MOVL    #10, ENUM_LST_SIZE                      ; 2221
                                  0204  31  00AE1 79$:      BRW     103$                                    ; 1586
                          0094  CE      D5  00AE4 80$:      TSTL    ENUM_LST_PTR                            ; 2231
                                        11  12 00AE8        BNEQ    81$
                                        57  DD 00AEA        PUSHL   MODNAMEPTR                              ; 2233
                                        01  DD 00AEC        PUSHL   #1
                          0002896A      8F  DD 00AEE        PUSHL   #166250
                00000000G   00          03  FB 00AF4        CALLS   #3, LIB$SIGNAL
                                        07  DD 00AFB 81$:    PUSHL   #7                                      ; 2235
                00000000G   00          01  FB 00AFD        CALLS   #1, DBG$GET_MEMORY
                            5A          50  D0 00B04        MOVL    R0, RSTPTR
                    14      AA          06  90 00B07        MOVB    #6, 20(RSTPTR)                          ; 2236
                    18      AA    70    AE  D0 00B0B        MOVL    ENUM_DSTPTR, 24(RSTPTR)                 ; 2237
                    15      AA          08  88 00B10        BISB2   #8, 21(RSTPTR)                          ; 2238
                    14      AE          01  D0 00B14        MOVL    #1, NEWENT_FLAG                         ; 2239
                                        5A  DD 00B18        PUSHL   RSTPTR                                  ; 2240
                          0094  CE      9F  00B1A           PUSHAB  ENUM_LST_SIZE
                          009C  CE      9F  00B1E           PUSHAB  ENUM_LST_PTR
                                  00FE  31  00B22           BRW     91$
                            52    0094  CE  D0 00B25 82$:    MOVL    ENUM_LST_PTR, R2                        ; 2251
                                        11  12 00B2A        BNEQ    83$
                                        57  DD 00B2C        PUSHL   MODNAMEPTR                              ; 2253
                                        01  DD 00B2E        PUSHL   #1
                          0002896A      8F  DD 00B30        PUSHL   #166250
                00000000G   00          03  FB 00B36        CALLS   #3, LIB$SIGNAL
             7E             62          0B  C1 00B3D 83$:    ADDL3   #11, (R2), -(SP)                       ; 2255
                00000000G   00          01  FB 00B41        CALLS   #1, DBG$GET_MEMORY
                            5A          50  D0 00B48        MOVL    R0, RSTPTR
                    14      AA          07  90 00B4B        MOVB    #7, 20(RSTPTR)                          ; 2256
                    18      AA          04  90 00B4F        MOVB    #4, 24(RSTPTR)                          ; 2257
                    20      AA    0084  CE  D0 00B53        MOVL    ENUM_BITSIZE, 32(RSTPTR)                ; 2258
                    2C      AE      70  AE  D0 00B59        MOVL    ENUM_DSTPTR, DSTPTR_TO_SAVE             ; 2259
                    14      AE          01  D0 00B5E        MOVL    #1, NEWENT_FLAG                         ; 2260
                    28      AA          62  D0 00B62        MOVL    (R2), 40(RSTPTR)                        ; 2261
                    4C      AE    2C    AA  9E 00B66        MOVAB   44(R10), ENUMELT                        ; 2262
                                        50  D4 00B6B        CLRL    I                                       ; 2264
                                        0A  11 00B6D        BRB     85$
             51     4C      AE          04  C3 00B6F 84$:    SUBL3   #4, ENUMELT, R1
                          6140          6240  D0 00B74        MOVL    (R2)[I], (R1)[I]
             F2           50            62  F3 00B79 85$:    AOBLEQ  (R2), I, 84$
                                        52  DD 00B7D        PUSHL   R2                                      ; 2266
                00000000G   00          01  FB 00B7F        CALLS   #1, DBG$REL_MEMORY
                                  0094  CE  D4 00B86        CLRL    ENUM_LST_PTR                            ; 2267
                                  015B  31  00B8A           BRW     103$                                    ; 1586
```

```
                                                07 DD 00B8D 86$:    PUSHL   #7                                          ; 2284
                                00000000G   00  01 FB 00B8F         CALLS   #1, DBG$GET_MEMORY
                                            5A  50 D0 00B96         MOVL    R0, RSTPTR
                                         14 AA  06 90 00B99         MOVB    #6, 20(RSTPTR)                              ; 2285
                                       18 AA    D4 00B9D         CLRL    24(RSTPTR)                                  ; 2286
                                         14 AE  01 D0 00BA0         MOVL    #1, NEWENT_FLAG                             ; 2287
                                            17  01 AB 91 00BA4         CMPB    1(R11), #23                             ; 2295
                                                19 12 00BA8         BNEQ    87$
                                            01  02 AB 91 00BAA         CMPB    2(R11), #1                              ; 2296
                                                13 12 00BAE         BNEQ    87$
                        00000000'  EF       07  11 29 00BB0         CMPC3   #17, 7(R11), P.AAK                      ; 2297
                                                08 12 00BB9         BNEQ    87$
                                00000000G   00  03 AB D0 00BBB         MOVL    3(R11), DBG$GL_TRANSFER_ADDRESS             ; 2300
                                         A3 8F  01 AB 91 00BC3 87$:     CMPB    1(R11), #163                            ; 2309
                                                29 12 00BC8         BNEQ    89$
                                          0098  CE 9F 00BCA         PUSHAB  ARRAY_FLAG                                  ; 2312
                                          00A0  CE 9F 00BCE         PUSHAB  TYPEPTR
                                                57 DD 00BD2         PUSHL   MODNAMEPTR
                                                5B DD 00BD4         PUSHL   R11
                                         0000V  CF  04 FB 00BD6         CALLS   #4, GET_TYPEPTR
                                         18 AA      009C  CE D0 00BDB         MOVL    TYPEPTR, 24(RSTPTR)             ; 2313
                                                08 13 00BE1         BEQL    88$                                         ; 2314
                                         15 AA      08 88 00BE3         BISB2   #8, 21(RSTPTR)                          ; 2317
                                         6C AE      01 D0 00BE7         MOVL    #1, BUILD_TYPE_TABLE                    ; 2318
                15 AA                 01            04  0098  CE F0 00BEB 88$:     INSV    ARRAY_FLAG, #4, #1, 21(RSTPTR)  ; 2321
                                                00A4  CE D5 00BF3 89$:     TSTL    COMP_LST_PTR                        ; 2328
                                                20 12 00BF7         BNEQ    90$
                                          00A8  CE 9F 00BF9         PUSHAB  STATIC_FLAG                                 ; 2331
                                          00B0  CE 9F 00BFD         PUSHAB  LENGTH
                                          00B8  CE 9F 00C01         PUSHAB  ADDRESS
                                                5B DD 00C05         PUSHL   R11
                                         0000V  CF  04 FB 00C07         CALLS   #4, STATIC_CHECK
                                            43      00A8  CE E9 00C0C         BLBC    STATIC_FLAG, 93$                ; 2332
                                                00AC  CE D5 00C11         TSTL    LENGTH                              ; 2335
                                                48 12 00C15         BNEQ    94$
                                                4A 11 00C17         BRB     95$                                     ; 2337
                                                5A DD 00C19 90$:    PUSHL   RSTPTR                                  ; 2348
                                          00A4  CE 9F 00C1B         PUSHAB  COMP_LST_SIZE
                                          00AC  CE 9F 00C1F         PUSHAB  COMP_LST_PTR
                                         0000V  CF  03 FB 00C23 91$:     CALLS   #3, COMP_LIST_ADD_ENTRY
                                                68 11 00C28         BRB     99$                                     ; 1586
                                                07 DD 00C2A 92$:    PUSHL   #7                                      ; 2359
                                00000000G   00  01 FB 00C2C         CALLS   #1, DBG$GET_MEMORY
                                            5A  50 D0 00C33         MOVL    R0, RSTPTR
                                         14 AA  06 90 00C36         MOVB    #6, 20(RSTPTR)                          ; 2360
                                       18 AA    D4 00C3A         CLRL    24(RSTPTR)                                  ; 2361
                                         14 AE  01 D0 00C3D         MOVL    #1, NEWENT_FLAG                         ; 2362
                                          00A8  CE 9F 00C41         PUSHAB  STATIC_FLAG                             ; 2363
                                          00B0  CE 9F 00C45         PUSHAB  LENGTH
                                          00B8  CE 9F 00C49         PUSHAB  ADDRESS
                                                5B DD 00C4D         PUSHL   R11
                                         0000V  CF  04 FB 00C4F         CALLS   #4, STATIC_CHECK
                                            39      00A8  CE E9 00C54 93$:     BLBC    STATIC_FLAG, 99$            ; 2364
                                                00AC  CE D5 00C59         TSTL    LENGTH                          ; 2367
                                                04 13 00C5D         BEQL    95$
                                         15 AA      02 88 00C5F 94$:    BISB2   #2, 21(RSTPTR)
                                                7E D4 00C63 95$:    CLRL    -(SP)                                   ; 2369
```

```
                                20   AE  DD 00C65          PUSHL   32(SP)                                          ;
                                5A   DD 00C68          PUSHL   RSTPTR                                          ;
                            00B8     CE  DD 00C6A          PUSHL   LENGTH                                          ;
                            00C0     CE  DD 00C6E          PUSHL   ADDRESS                                         ;
                    0000V   CF       05  FB 00C72 96$:     CALLS   #5, STATIC_ADDRESS                              ;
                                     6F  11 00C77          BRB     103$                                            ; 1586
                                     07  DD 00C79 97$:     PUSHL   #7                                              ; 2390
                00000000G   00       01  FB 00C7B          CALLS   #1, DBG$GET_MEMORY                              ;
                            5A       50  D0 00C82          MOVL    R0, RSTPTR                                      ;
                    14      AA       06  90 00C85          MOVB    #6, 20(RSTPTR)                                  ; 2391
                                18   AA  D4 00C89          CLRL    24(RSTPTR)                                      ; 2392
                                     56  11 00C8C          BRB     102$                                            ; 2393
                    68      AE       01  D0 00C8E 98$:     MOVL    #1, COBOLGBL_FLAG                               ; 2403
                                     54  11 00C92 99$:     BRB     103$                                            ;
                                     06  DD 00C94 100$:    PUSHL   #6                                              ; 2410
                00000000G   00       01  FB 00C96          CALLS   #1, DBG$GET_MEMORY                              ;
                            5A       50  D0 00C9D          MOVL    R0, RSTPTR                                      ;
                    14      AA       0D  90 00CA0          MOVB    #13, 20(RSTPTR)                                 ; 2411
                                     3E  11 00CA4          BRB     102$                                            ; 2412
                            51   02  AB  9E 00CA6 101$:     MOVAB   2(R11), R1                                      ; 2424
                            50       61  9A 00CAA          MOVZBL  (R1), R0                                        ;
                                     39  13 00CAD          BEQL    103$                                            ;
                    40      AE   01 A140 9E 00CAF          MOVAB   1(R1)[R0], TSPTR                                ; 2430
                                     0B  DD 00CB5          PUSHL   #11                                             ; 2434
                00000000G   00       01  FB 00CB7          CALLS   #1, DBG$GET_MEMORY                              ;
                            5A       50  D0 00CBE          MOVL    R0, RSTPTR                                      ;
                    0C      AA       5B  D0 00CC1          MOVL    R11, 12(RSTPTR)                                 ; 2435
                    10      AA 00000000' EF  D0 00CC5          MOVL    RST$START_ADDR, 16(RSTPTR)                      ; 2436
                    14      AA       07  90 00CCD          MOVB    #7, 20(RSTPTR)                                  ; 2437
                52  40      AE       02  C1 00CD1          ADDL3   #2, TSPTR, R2                                   ; 2438
                            7E       62  9A 00CD6          MOVZBL  (R2), -(SP)                                     ;
                00000000G   00       01  FB 00CD9          CALLS   #1, DBG$TRANS_TYPE_CODE                         ;
                    18      AA       50  90 00CE0          MOVB    R0, 24(RSTPTR)                                  ;
                    14      AE       01  D0 00CE4 102$:     MOVL    #1, NEWENT_FLAG                                 ; 2451
                            2E   14  AE  E9 00CE8 103$:     BLBC    NEWENT_FLAG, 105$                               ; 2469
                    0C      AA   2C  AE  D0 00CEC          MOVL    DSTPTR_TO_SAVE, 12(RSTPTR)                      ; 2472
                            50   04  AE  D0 00CF1          MOVL    NESTING, R0                                     ; 2473
                    10      AA FF38 CD40 D0 00CF5          MOVL    NEST_STACK[R0], 16(RSTPTR)                      ;
                            04   68  AE  E9 00CFC          BLBC    COBOLGBL_FLAG, 104$                             ; 2474
                    15      AA       20  88 00D00          BISB2   #32, 21(RSTPTR)                                 ;
                                 68  AE  D4 00D04 104$:     CLRL    COBOLGBL_FLAG                                   ; 2475
                50  24      AE       08  C1 00D07          ADDL3   #8, SYMCHNPTR, R0                               ; 2476
                            60       5A  D0 00D0C          MOVL    RSTPTR, (R0)                                    ;
                    24      AE       5A  D0 00D0F          MOVL    RSTPTR, SYMCHNPTR                               ; 2477
                                     5A  DD 00D13          PUSHL   RSTPTR                                          ; 2486
                    F264    CF       01  FB 00D15          CALLS   #1, DBG$HASH_INSERT                             ;
                            24   60  AE  E9 00D1A 105$:     BLBC    NEST_FLAG, 107$                                 ; 2494
                            31   04  AE  D1 00D1E          CMPL    NESTING, #49                                    ; 2497
                                     11  19 00D22          BLSS    106$                                            ;
                                     57  DD 00D24          PUSHL   MODNAMEPTR                                      ; 2499
                                     01  DD 00D26          PUSHL   #1                                              ;
                                00028962 8F  DD 00D28          PUSHL   #166242                                         ;
                00000000G   00       03  FB 00D2E          CALLS   #3, LIB$SIGNAL                                  ;
                                 04  AE  D6 00D35 106$:     INCL    NESTING                                         ; 2501
                            50   04  AE  D0 00D38          MOVL    NESTING, R0                                     ; 2502
                    FF38 CD40            5A  D0 00D3C          MOVL    RSTPTR, NEST_STACK[R0]                          ;
                            19   54  AE  E9 00D42 107$:     BLBC    UNNEST_FLAG, 109$                               ; 2509
```

```
                        04    AE   D5 00D46           TSTL    NESTING                                 ; 2512
                              11   14 00D49           BGTR    108$                                    ;
                              57   DD 00D4B           PUSHL   MODNAMEPTR                              ;
                              01   DD 00D4D           PUSHL   #1                                      ;
                        0002896A   8F   DD 00D4F           PUSHL   #166250                                 ;
            00000000G   00             03   FB 00D55           CALLS   #3, LIB$SIGNAL                          ;
                              04    AE   D7 00D5C 108$:      DECL    NESTING                                 ; 2513
                 008C   CE      0088   CE   D0 00D5F 109$:      MOVL    NEXT_DSTPTR, DSTPTR                     ; 2520
                           F3BD   31   00D66           BRW     7$                                      ; 1553
                              59   D5 00D69 110$:      TSTL    NUM_PC_TBLS                             ; 2531
                              37   15 00D6B           BLEQ    113$                                    ;
          50        04    AC             1C   C1 00D6D           ADDL3   #28, MODRSTPTR, R0                      ; 2534
                        60      5C   AE   D0 00D72           MOVL    PCTBL_BASE, (R0)                        ;
                              01    A9   9F 00D76           PUSHAB  1(NUM_PC_TBLS)                          ; 2535
            00000000G   00             01   FB 00D79           CALLS   #1, DBG$GET_MEMORY                      ;
                 28    AE             50   D0 00D80           MOVL    R0, MOD_PC_TBL                          ;
          50        04    AC             2C   C1 00D84           ADDL3   #44, MODRSTPTR, R0                      ; 2536
                        60      28   AE   D0 00D89           MOVL    MOD_PC_TBL, (R0)                        ;
                 28    BE             59   D0 00D8D           MOVL    NUM_PC_TBLS, @MOD_PC_TBL                ; 2537
                              50   D4 00D91           CLRL    I                                       ; 2539
                              0B   11 00D93           BRB     112$                                    ;
          51        34    AE             04   C3 00D95 111$:      SUBL3   #4, PC_CORR_TBL, R1                     ;
                 28 BE40             6140   D0 00D9A           MOVL    (R1)[I], @MOD_PC_TBL[I]                 ;
          F1              50             59   F3 00DA0 112$:      AOBLEQ  NUM_PC_TBLS, I, 111$            ;
                              34    AE   DD 00DA4 113$:      PUSHL   PC_CORR_TBL                             ; 2543
            00000000G   00             01   FB 00DA7           CALLS   #1, DBG$REL_MEMORY                      ;
                              58   D5 00DAE           TSTL    NUM_SRC_DSTS                            ; 2552
                              2E   13 00DB0           BEQL    116$                                    ;
                              01    A8   9F 00DB2           PUSHAB  1(NUM_SRC_DSTS)                         ; 2555
               00000000G   00             01   FB 00DB5           CALLS   #1, DBG$GET_MEMORY                      ;
                        64    AE             50   D0 00DBC           MOVL    R0, MOD_SRC_TBL                         ;
          50        04    AC             24   C1 00DC0           ADDL3   #36, MODRSTPTR, R0                      ; 2556
                        60      64   AE   D0 00DC5           MOVL    MOD_SRC_TBL, (R0)                       ;
                 64    BE             58   D0 00DC9           MOVL    NUM_SRC_DSTS, @MOD_SRC_TBL              ; 2557
                              50   D4 00DCD           CLRL    I                                       ; 2559
                              0B   11 00DCF           BRB     115$                                    ;
          51        30    AE             04   C3 00DD1 114$:      SUBL3   #4, SRC_CORR_TBL, R1                    ;
                 64 BE40             6140   D0 00DD6           MOVL    (R1)[I], @MOD_SRC_TBL[I]                ;
          F1              50             58   F3 00DDC 115$:      AOBLEQ  NUM_SRC_DSTS, I, 114$           ;
                              30    AE   DD 00DE0 116$:      PUSHL   SRC_CORR_TBL                            ; 2563
            00000000G   00             01   FB 00DE3           CALLS   #1, DBG$REL_MEMORY                      ;
                              03             6E   E8 00DEA           BLBS    CONTINUATION_DST_SEEN, 117$             ; 2571
                           0080   31   00DED           BRW     126$                                    ;
          50        04    AC             08   C1 00DF0 117$:      ADDL3   #8, MODRSTPTR, R0                       ; 2578
                        5A              60   D0 00DF5           MOVL    (R0), RSTPTR                            ;
                              76   13 00DF8 118$:      BEQL    126$                                    ; 2579
                              0C    AA   DD 00DFA           PUSHL   12(RSTPTR)                              ; 2585
                 0000V  CF             01   FB 00DFD           CALLS   #1, DBG$RST_DST_PTR                     ;
                        0C    AA             50   D0 00E02           MOVL    R0, 12(RSTPTR)                          ;
                              50   14    AA   9E 00E06           MOVAB   20(RSTPTR), R0                          ; 2590
                              06             50   D1 00E0A           CMPL    R0, #6                                  ;
                              0C   12 00E0D           BNEQ    119$                                    ;
                              18    AA   DD 00E0F           PUSHL   24(RSTPTR)                              ; 2592
                 0000V  CF             01   FB 00E12           CALLS   #1, DBG$RST_DST_PTR                     ;
                        18    AA             50   D0 00E17           MOVL    R0, 24(RSTPTR)                          ;
                              50   14    AA   9E 00E1B 119$:      MOVAB   20(RSTPTR), R0                          ; 2597
                              07             50   D1 00E1F           CMPL    R0, #7                                  ;
```

```
                                    0:  12 00E22          BNEQ    120$                                    ;
                                24  AA  DD 00E24          PUSHL   36(RSTPTR)                              ; 2600
                          0000V CF      0' FB 00E27          CALLS   #1, DBG$RST_DST_PTR                     ;
                             24 AA      50 D0 00E2C          MOVL    R0, 36(RSTPTR)                          ;
                                07  14  AA 91 00E30 120$:    CMPB    20(RSTPTR), #7                          ; 2605
                                34  12 00E34          BNEQ    125$                                    ;
                                59  2C  AA 9E 00E36          MOVAB   44(R10), TYPCOMPLST                     ; 2612
                                55      01 CE 00E3A          MNEGL   #1, J                                   ; 2619
                                26  11 00E3D          BRB     124$                                    ;
                                52    6945 D0 00E3F 121$:    MOVL    (TYPCOMPLST)[J], RST_PTR                ;
                                0B  14  A2 91 00E43          CMPB    20(RST_PTR), #11                        ; 2620
                                1C  12 00E47          BNEQ    124$                                    ;
                                53  18  A2 9E 00E49          MOVAB   24(R2), VARSETTBL                       ; 2627
                                54      01 CE 00E4D          MNEGL   #1, I                                   ; 2630
                                0E  11 00E50          BRB     123$                                    ;
                                56    6344 D0 00E52 122$:    MOVL    (VARSETTBL)[I], VARIANT_ENTRY           ;
                                66  DD 00E56          PUSHL   (VARIANT_ENTRY)                         ; 2632
                          0000V CF      01 FB 00E58          CALLS   #1, DBG$RST_DST_PTR                     ;
                                66      50 D0 00E5D          MOVL    R0, (VARIANT_ENTRY)                     ;
ED                              54  08  A2 F2 00E60 123$:    AOBLSS  8(RST_PTR), I, 122$                     ; 2628
D5                              55  28  AA F2 00E65 124$:    AOBLSS  40(RSTPTR), J, 121$                     ; 2613
                                5A  08  AA D0 00E6A 125$:    MOVL    8(RSTPTR), RSTPTR                       ; 2644
                                88  11 00E6E          BRB     118$                                    ; 2579
                                03  6C  AE E8 00E70 126$:    BLBS    BUILD_TYPE_TABLE, 127$                  ; 2657
                                  0187  31 00E74          BRW     149$                                    ;
                                    0B  DD 00E77 127$:    PUSHL   #11                                     ; 2661
                    00000000G   00      01 FB 00E79          CALLS   #1, DBG$GET_MEMORY                      ;
                        00B8    CE      50 D0 00E80          MOVL    R0, TYPE_DST_LST                        ;
                        00B4    CE      0A D0 00E85          MOVL    #10, TYPE_DST_LEN                       ; 2662
                                    0B  DD 00E8A          PUSHL   #11                                     ; 2663
                    00000000G   00      01 FB 00E8C          CALLS   #1, DBG$GET_MEMORY                      ;
                        00C0    CE      50 D0 00E93          MOVL    R0, TYPE_RST_LST                        ;
                        00BC    CE      0A D0 00E98          MOVL    #10, TYPE_RST_LEN                       ; 2664
50                        04    AC      08 C1 00E9D          ADDL3   #8, MODRSTPTR, R0                       ; 2665
                                5A      60 D0 00EA2          MOVL    (R0), RSTPTR                            ;
                                    2B  13 00EA5 128$:    BEQL    130$                                    ; 2666
                                07      14  AA 91 00EA7          CMPB    20(RSTPTR), #7                          ; 2668
                                    1F  12 00EAB          BNEQ    129$                                    ;
                                        0C  AA DD 00EAD          PUSHL   12(RSTPTR)                              ; 2672
                                    00B8    CE 9F 00EB0          PUSHAB  TYPE_DST_LEN                            ; 2671
                                    00C0    CE 9F 00EB4          PUSHAB  TYPE_DST_LST                            ;
                          0000V CF          03 FB 00EB8          CALLS   #3, COMP_LIST_ADD_ENTRY                 ;
                                5A      DD 00EBD          PUSHL   RSTPTR                                  ; 2673
                                    00C0    CE 9F 00EBF          PUSHAB  TYPE_RST_LEN                            ;
                                    00C8    CE 9F 00EC3          PUSHAB  TYPE_RST_LST                            ;
                          0000V CF          03 FB 00EC7          CALLS   #3, COMP_LIST_ADD_ENTRY                 ;
                                5A      08  AA D0 00ECC 129$:    MOVL    8(RSTPTR), RSTPTR                       ; 2676
                                    D3  11 00ED0          BRB     128$                                    ; 2666
50                        04    AC      08 C1 00ED2 130$:    ADDL3   #8, MODRSTPTR, R0                       ; 2684
                                5A      60 D0 00ED7          MOVL    (R0), RSTPTR                            ;
                                    41  13 00EDA 131$:    BEQL    136$                                    ; 2685
36                        15    AA      03 E1 00EDC          BBC     #3, 21(RSTPTR), 135$                    ; 2687
                                    50  D4 00EE1          CLRL    I                                       ; 2692
                                    16  11 00EE3          BRB     133$                                    ;
                                18  AA  00B8 DE40 D1 00EE5 132$:    CMPL    @TYPE_DST_LST[I], 24(RSTPTR)            ;
                                        0D  12 00EEC          BNEQ    133$                                    ;
                                18  AA  00C0 DE40 D0 00EEE          MOVL    @TYPE_RST_LST[I], 24(RSTPTR)            ; 2695
```

```
                15   AA            09  8A 00EF5          BICB2   #8, 21(RSTPTR)                          ; 2696
                                   06  11 00EF9          BRB     134$                                    ; 2694
E4                   50      00B8  DE  F3 00EFB 133$:    AOBLEQ  @TYPE_DST_LST, I, 132$                  ; 2690
11              15   AA            03  E1 00F01 134$:    BBC     #3, 21(RSTPTR), 135$                    ; 2702
                                   57  DD 00F06          PUSHL   MODNAMEPTR                              ; 2704
                                   01  DD 00F08          PUSHL   #1
                     0002896A      8F  DD 00F0A          PUSHL   #166250
    00000000G        00            03  FB 00F10          CALLS   #3, LIB$SIGNAL
                     5A      08    AA  D0 00F17 135$:    MOVL    8(RSTPTR), RSTPTR                       ; 2708
                                   BD  11 00F1B          BRB     131$                                    ; 2685
50              04   AC            08  C1 00F1D 136$:    ADDL3   #8, MODRSTPTR, R0                       ; 2719
                     5A            60  D0 00F22          MOVL    (R0), RSTPTR
                                   46  13 00F25 137$:    BEQL    141$                                    ; 2720
                     06      14    AA  91 00F27          CMPB    20(RSTPTR), #6                          ; 2722
                                   06  13 00F2B          BEQL    138$
                     0A      14    AA  91 00F2D          CMPB    20(RSTPTR), #10                         ; 2723
                                   34  12 00F31          BNEQ    140$
                             18    AA  D5 00F33 138$:    TSTL    24(RSTPTR)                              ; 2726
                                   2F  13 00F36          BEQL    140$
                009C         CE    18  AA  D0 00F38          MOVL    24(RSTPTR), TYPEPTR                     ; 2729
                             52  009C  CE  D0 00F3E          MOVL    TYPEPTR, R2                             ; 2730
                             07    14  A2  91 00F43          CMPB    20(R2), #7
                                   15  13 00F47          BEQL    139$
                     00000000'     EF  9F 00F49          PUSHAB  P.AAL                                   ; 2732
                                   01  DD 00F4F          PUSHL   #1
                     00028362      8F  DD 00F51          PUSHL   #164706
    00000000G        00            03  FB 00F57          CALLS   #3, LIB$SIGNAL
                             07    18  A2  91 00F5E 139$:    CMPB    24(R2), #7                              ; 2734
                                   03  12 00F62          BNEQ    140$
                           1A      A2  B6 00F64          INCW    26(R2)                                  ; 2736
                     5A      08    AA  D0 00F67 140$:    MOVL    8(RSTPTR), RSTPTR                       ; 2742
                                   B8  11 00F6B          BRB     137$                                    ; 2720
                                   53  D4 00F6D 141$:    CLRL    I                                       ; 2752
                                   24  11 00F6F          BRB     143$
                009C         CE  00C0 DE43  D0 00F71 142$:    MOVL    @TYPE_RST_LST[I], TYPEPTR               ; 2754
                             52  009C  CE  D0 00F79          MOVL    TYPEPTR, R2                             ; 2755
                               1A    A2  B5 00F7E          TSTW    26(R2)
                                   12  13 00F81          BEQL    143$
                     7E        1A    A2  3C 00F83          MOVZWL  26(R2), -(SP)                           ; 2758
    00000000G        00            01  FB 00F87          CALLS   #1, DBG$GET_MEMORY
                   1C      A2            50  D0 00F8E          MOVL    R0, 28(R2)
                             1A    A2  B4 00F92          CLRW    26(R2)                                  ; 2759
D6                   53      00C0  DE  F3 00F95 143$:    AOBLEQ  @TYPE_RST_LST, I, 142$                  ; 2752
50              04   AC            08  C1 00F9B          ADDL3   #8, MODRSTPTR, R0                       ; 2772
                     5A            60  D0 00FA0          MOVL    (R0), RSTPTR
                                   43  13 00FA3 144$:    BEQL    148$                                    ; 2773
                     06      14    AA  91 00FA5          CMPB    20(RSTPTR), #6                          ; 2775
                                   06  13 00FA9          BEQL    145$
                     0A      14    AA  91 00FAB          CMPB    20(RSTPTR), #10                         ; 2776
                                   31  12 00FAF          BNEQ    147$
                             18    AA  D5 00FB1 145$:    TSTL    24(RSTPTR)                              ; 2779
                                   2C  13 00FB4          BEQL    147$
                009C         CE    18  AA  D0 00FB6          MOVL    24(RSTPTR), TYPEPTR                     ; 2788
                             50  009C  CE  D0 00FBC          MOVL    TYPEPTR, R0                             ; 2789
                             07    18  A0  91 00FC1          CMPB    24(R0), #7
                                   0F  12 00FC5          BNEQ    146$
                             52    1C  A0  D0 00FC7          MOVL    28(R0), TYPREFTBL                       ; 2792
```

```
                          51        1A    A0  3C 00FCB           MOVZWL  26(R0), R1                                    ; 2793
                        6241              5A  D0 00FCF           MOVL    RSTPTR, (TYPREFTBL)[R1]                         ;
                                    1A    A0  B6 00FD3           INCW    26(R0)                                          ; 2794
                07        15    AA        04  E1 00FD6 146$:     BBC     #4, 21(RSTPTR), 147$                            ; 2803
                                    18    AA  D4 00FDB           CLRL    24(RSTPTR)                                      ; 2806
                          15    AA        10  8A 00FDE           BICB2   #16, 21(RSTPTR)                                 ; 2807
                                5A        08  AA  D0 00FE2 147$:  MOVL    8(RSTPTR), RSTPTR                               ; 2814
                                          BB  11 00FE6           BRB     146$                                            ; 2773
                                    00B8  CE  DD 00FE8 148$:     PUSHL   TYPE_DST_LST                                    ; 2820
                00000000G       00        01  FB 00FEC           CALLS   #1, DBG$REL_MEMORY                              ;
                                    00C0  CE  DD 00FF3           PUSHL   TYPE_RST_LST                                    ; 2821
                00000000G       00        01  FB 00FF7           CALLS   #1, DBG$REL_MEMORY                              ;
                          50    BE        02  88 00FFE 149$:     BISB2   #2, @80(SP)                                     ; 2827
                                          04  01002              RET                                                     ; 2829
                                        0000  01003 150$:        .WORD   Save nothing                                    ; 1294
                                          7E  D4 01005           CLRL    -(SP)                                           ;
                                          5E  DD 01007           PUSHL   SP                                              ;
                                7E        04  AC  7D 01009      MOVQ    4(AP), -(SP)                                    ;
                      0000V     CF        03  FB 0100D           CALLS   #3, RST_BUILD_ERROR_HANDLER                     ;
                                          04  01012              RET                                                     ;

; Routine Size:  4115 bytes,    Routine Base:  DBG$CODE + 07BE
```

```
;2707     2830  1 GLOBAL ROUTINE DBG$RST_BLDSCOPE_LIST(PNVEC, ERRINDEX,
;2708     2831  1                                      ORIGIN_FLAG, MODULE_FLAG): =
;2709     2832  1 !
;2710     2833  1 ! FUNCTION
;2711     2834  1 !       This routine makes a new set of scopes available to the symbol table
;2712     2835  1 !       access routines.  It is thus called in the course of processing the
;2713     2836  1 !       set scope list.  It accepts as input a vector of pointers to
;2714     2837  1 !       Pathname Descriptors for the scopes specified on the SET SCOPE command,
;2715     2838  1 !       or SET SYMBOL/qulifiers namespec IN scopespec command.  It builds a
;2716     2839  1 !       linked list of the corresponding Scope Entries, and makes a scope
;2717     2840  1 !       pointer point to this list.
;2718     2841  1 !
;2719     2842  1 !       If an error in a scope pathname is detected, the index of that pathname
;2720     2843  1 !       is returned to ERRINDEX.  The actual return is accomplished by signal-
;2721     2844  1 !       ling an error which is intercepted by an error handler.  That handler
;2722     2845  1 !       releases any partially built new scope list back to the free memory pool
;2723     2846  1 !       and then unwinds to return control to the caller of DBG$RST_SETSCOPE,
;2724     2847  1 !       or DBG$STA_SHOWSYMBOL.
;2725     2848  1 !
;2726     2849  1 ! INPUTS
;2727     2850  1 !       PNVEC    - A vector of pointers to Pathname Descriptors for the scopes
;2728     2851  1 !                  specified on the SET SCOPE command or SHOW SYMBOL command.
;2729     2852  1 !                  PNVEC[0] gives the number of such pointers and PNVEC[1]
;2730     2853  1 !                  through PNVEC[,PNVEC[0]] contain the pointers to the
;2731     2854  1 !                  Pathname Descriptors.  The CANCEL SCOPE command is handled
;2732     2855  1 !                  by making PNVEC[0] zero.
;2733     2856  1 !
;2734     2857  1 !       ERRINDEX - The address of a longword location to receive an "error
;2735     2858  1 !                  index"; see description below.
;2736     2859  1 !
;2737     2860  1 !       ORIGIN_FLAG - A flag set to TRUE if this routine is called from
;2738     2861  1 !                     DBG$RST_SETSCOPE. This prevents us from setting the
;2739     2862  1 !                     module as a side effect of the command.
;2740     2863  1 !       MODULE_FLAG - TRUE if we were called from SET SCOPE/MODULE
;2741     2864  1 !
;2742     2865  1 ! OUTPUTS
;2743     2866  1 !       ERRINDEX - If some pathname is found to be invalid, the PNVEC index of
;2744     2867  1 !                  that pathname is returned to ERRINDEX.  The calling routine
;2745     2868  1 !                  can then use ERRINDEX to produce an error message.  If all
;2746     2869  1 !                  scope pathnames are correct, zero is returned to ERRINDEX.
;2747     2870  1 !
;2748     2871  1 !       NEW_SCOPE_LIST - Return value from this routine which is the pointer
;2749     2872  1 !                        point to the newly build scope list.
;2750     2873  1 !                                                                              ;
;2751     2874  1
;2752     2875  2     BEGIN
;2753     2876  2
;2754     2877  2     MAP
;2755     2878  2         PNVEC: REF VECTOR[,LONG],       ! Pointer to vector of Pathname Descrip-
;2756     2879  2                                         !     tor pointers
;2757     2880  2         ERRINDEX: REF VECTOR[1];        ! Pointer to error index longword
;2758     2881  2
;2759     2882  2     LOCAL
;2760     2883  2         KIND,                           ! The Kind of the scope's RST entry
;2761     2884  2         MODPTR: REF RST$ENTRY,          ! Pointer to Module RST Entry for module
;2762     2885  2                                         !     containing the present scope
;2763     2886  2         NEW_SCOPE_LIST: VOLATILE,       ! Pointer to new scope list being built
```

```
;2764    2887  2          PATHNAME: REF PTH$PATHNAME,      ! Pointer to current Pathname Descriptor
;2765    2888  2          PATHVEC: REF VECTOR[,LONG],      ! Pointer to current pathname vector
;2766    2889  2          PATH_STRING,                     ! Pointer to scope name as counted ASCII
;2767    2890  2          PNAME: REF VECTOR[,BYTE],        ! Pointer to first name in the Pathname
;2768    2891  2                                           !     Descriptor (as Counted ASCII)
;2769    2892  2          PREVSCOPE: REF SCOPE$ENTRY,      ! Pointer to the scope entry before the
;2770    2893  2                                           !     present scope entry
;2771    2894  2          RSTPTR: REF RST$ENTRY,           ! Pointer to current scope's RST entry
;2772    2895  2          SCOPE: REF SCOPE$ENTRY,          ! Pointer to the present scope entry
;2773    2896  2          STATUS;                          ! Status code returned by other routines
;2774    2897  2
;2775    2898  2      ENABLE
;2776    2899  2          BLDSCOPE_ERROR_HANDLER(NEW_SCOPE_LIST); ! Declare an error handler
;2777    2900  2
;2778    2901  2
;2779    2902  2
;2780    2903  2      ! Make NEW_SCOPE_LIST point to the list of new scopes.
;2781    2904  2      !
;2782    2905  2      PREVSCOPE = NEW_SCOPE_LIST;
;2783    2906  2      PREVSCOPE[SCOPE$L_FLINK] = 0;
;2784    2907  2
;2785    2908  2
;2786    2909  2      ! If PNVEC is empty (as for CANCEL SCOPE), we build a scope list entry for
;2787    2910  2      ! numbered scope "0", i.e. for the scope containing the current PC.
;2788    2911  2      !
;2789    2912  2      IF .PNVEC[0] EQL 0
;2790    2913  2      THEN
;2791    2914  3          BEGIN
;2792    2915  3          SCOPE = DBG$GET_MEMORY(SCOPE$K_ENTSIZE);
;2793    2916  3          SCOPE[SCOPE$L_FLINK] = 0;
;2794    2917  3          SCOPE[SCOPE$L_STATE] = SCOPE$K_NUMBERED;
;2795    2918  3          SCOPE[SCOPE$L_RSTPTR] = 0;
;2796    2919  3          SCOPE[SCOPE$L_MODPTR] = 0;
;2797    2920  3          PREVSCOPE[SCOPE$L_FLINK] = .SCOPE;
;2798    2921  3          PREVSCOPE = .SCOPE;
;2799    2922  2          END;
;2800    2923  2
;2801    2924  2
;2802    2925  2      ! Loop through the PNVEC vector and build a new scope entry for each element
;2803    2926  2      ! in that vector.
;2804    2927  2      !
;2805    2928  2      INCR I FROM 1 TO .PNVEC[0] DO
;2806    2929  3          BEGIN
;2807    2930  3          ERRINDEX[0] = .I;
;2808    2931  3
;2809    2932  3
;2810    2933  3          ! Set up pointers to the Pathname Descriptor, the pathname vector in
;2811    2934  3          ! that descriptor, to the first name in the pathname vector, and to the
;2812    2935  3          ! whole pathname string as Counted ASCII.
;2813    2936  3          !
;2814    2937  3          PATHNAME = .PNVEC[.I];
;2815    2938  3          PATHVEC = PATHNAME[PTH$A_PATHVECTOR];
;2816    2939  3          PNAME = .PATHVEC[0];
;2817    2940  3          DBG$NPATHDESC_TO_CS(.PATHNAME, PATH_STRING);
;2818    2941  3
;2819    2942  3
;2820    2943  3          ! Make sure there is no data qualification in the pathname.
```

```
;  2821     2944  3          !
;  2822     2945  3          IF .PATHNAME[PTH$B_TOTCNT] NEQ .PATHNAME[PTH$B_PATHCNT]
;  2823     2946  3          THEN
;  2824     2947  3              SIGNAL(DBG$_BADSCOPE, 1, .PATH_STRING);
;  2825     2948  3
;  2826     2949  3
;  2827     2950  3          ! If SET SCOPE/MODULE was specified, we process it here.
;  2828     2951  3          !
;  2829     2952  3          IF .MODULE_FLAG
;  2830     2953  3          THEN
;  2831     2954  4              BEGIN
;  2832     2955  4
;  2833     2956  4              ! Make sure there is exactly one name in the scope.
;  2834     2957  4              !
;  2835     2958  4              IF .PATHNAME[PTH$B_PATHCNT] NEQ 1
;  2836     2959  4              THEN
;  2837     2960  4                  SIGNAL(DBG$_MODUSCOPE, 1, .PATH_STRING);
;  2838     2961  4              IF .PNAME[0] EQL 0
;  2839     2962  4              THEN
;  2840     2963  4                  SIGNAL(DBG$_MODUSCOPE, 1, .PATH_STRING);
;  2841     2964  4
;  2842     2965  4
;  2843     2966  4              ! Find the module RST.
;  2844     2967  4              !
;  2845     2968  4              RSTPTR = GET_MOD_RST(PNAME[1], .PNAME[0]);
;  2846     2969  4
;  2847     2970  4
;  2848     2971  4              ! If the name is not a module, signal an error.
;  2849     2972  4              !
;  2850     2973  4              IF .RSTPTR EQL 0 THEN SIGNAL(DBG$_NOSUCHMODU, 1, .PATH_STRING);
;  2851     2974  4
;  2852     2975  4
;  2853     2976  4              ! If the module is not set, try to set it.
;  2854     2977  4              !
;  2855     2978  4              IF NOT .RSTPTR[RST$V_MODSET]
;  2856     2979  4              THEN
;  2857     2980  4                  IF NOT DBG$RST_SETMOD(PNAME[1], .PNAME[0], FALSE)
;  2858     2981  4                  THEN
;  2859     2982  4                      $DBG_ERROR('RSTCNTRL\RST_BLDSCOPE_LIST');
;  2860     2983  4
;  2861     2984  4
;  2862     2985  4              ! We have now found the scope in the RST.  Create a Normal scope
;  2863     2986  4              ! entry for it.
;  2864     2987  4              !
;  2865     2988  4              MODPTR = .RSTPTR;
;  2866     2989  4              WHILE .MODPTR[RST$B_KIND] NEQ RST$K_MODULE DO
;  2867     2990  4                  MODPTR = .MODPTR[RST$L_UPSCOPEPTR];
;  2868     2991  4
;  2869     2992  4              SCOPE = DBG$GET_MEMORY(SCOPE$K_ENTSIZE);
;  2870     2993  4              SCOPE[SCOPE$L_FLINK] = 0;
;  2871     2994  4              SCOPE[SCOPE$L_STATE] = SCOPE$K_NORMAL;
;  2872     2995  4              SCOPE[SCOPE$L_RSTPTR] = .RSTPTR;
;  2873     2996  4              SCOPE[SCOPE$L_MODPTR] = .MODPTR;
;  2874     2997  4              END
;  2875     2998  4
;  2876     2999  4
;  2877     3000  4          ! If the first name in the pathname is null, we either have a numbered
```

```
;  2878     3001  4          ! scope or the global scope.  Handle these two cases.
;  2879     3002  4          !
;  2880     3003  3          ELSE IF .PNAME[0] EQL 0
;  2881     3004  3          THEN
;  2882     3005  4              BEGIN
;  2883     3006  4              IF .PATHNAME[PTH$B_TOTCNT] NEQ 1
;  2884     3007  4              THEN
;  2885     3008  4                  SIGNAL(DBG$_BADSCOPE, 1, .PATH_STRING);
;  2886     3009  4
;  2887     3010  4
;  2888     3011  4              ! If this is the Global scope (i.e., "\"), build a Global Scope
;  2889     3012  4              ! Entry for inclusion on the new scope list.
;  2890     3013  4              !
;  2891     3014  4              IF .PATHNAME[PTH$B_LOCINVOC] EQL 0
;  2892     3015  4              THEN
;  2893     3016  5                  BEGIN
;  2894     3017  5                  SCOPE = DBG$GET_MEMORY(SCOPE$K_ENTSIZE);
;  2895     3018  5                  SCOPE[SCOPE$L_FLINK] = 0;
;  2896     3019  5                  SCOPE[SCOPE$L_STATE] = SCOPE$K_GLOBAL;
;  2897     3020  5                  SCOPE[SCOPE$L_RSTPTR] = 0;
;  2898     3021  5                  SCOPE[SCOPE$L_MODPTR] = 0;
;  2899     3022  5                  END
;  2900     3023  5
;  2901     3024  5
;  2902     3025  5              ! If this is a Numbered scope (i.e., "0"), build a Numbered Scope
;  2903     3026  5              ! Entry for inclusion on the new scope list.
;  2904     3027  5              !
;  2905     3028  4              ELSE IF .PATHNAME[PTH$B_LOCINVOC] EQL 1
;  2906     3029  4              THEN
;  2907     3030  5                  BEGIN
;  2908     3031  5                  SCOPE = DBG$GET_MEMORY(SCOPE$K_ENTSIZE);
;  2909     3032  5                  SCOPE[SCOPE$L_FLINK] = 0;
;  2910     3033  5                  SCOPE[SCOPE$L_STATE] = SCOPE$K_NUMBERED;
;  2911     3034  5                  SCOPE[SCOPE$L_RSTPTR] = 0;
;  2912     3035  5                  SCOPE[SCOPE$L_MODPTR] = .PATHNAME[PTH$L_INVOCNUM];
;  2913     3036  5                  END
;  2914     3037  5
;  2915     3038  5
;  2916     3039  5              ! Anything else with a null first name is an error.
;  2917     3040  5              !
;  2918     3041  4              ELSE
;  2919     3042  4                  SIGNAL(DBG$_BADSCOPE, 1, .PATH_STRING);
;  2920     3043  4
;  2921     3044  4              END
;  2922     3045  4
;  2923     3046  4
;  2924     3047  4          ! This seems to be an ordinary pathname.  Look it up in the RST.  If it
;  2925     3048  4          ! is not there, we see if it names a new module; if so, we SET that mod-
;  2926     3049  4          ! ule and look up the symbol in that module.
;  2927     3050  4          !
;  2928     3051  3          ELSE
;  2929     3052  4              BEGIN
;  2930     3053  4
;  2931     3054  4
;  2932     3055  4              ! Look up the pathname in the currently SET scopes and modules.  We
;  2933     3056  4              ! have a loop here since we may have to repeat the process if the
;  2934     3057  4              ! new scope is in a module which is not yet SET.
```

```
;  2935     3058  4          !
;  2936     3059  4          MODPTR = 0;
;  2937     3060  4          WHILE TRUE DO
;  2938     3061  5              BEGIN
;  2939     3062  5
;  2940     3063  5
;  2941     3064  5              ! Look up the symbol in the current scopes.  Then see what KIND
;  2942     3065  5              ! of symbol we got back.  We do not accept global (GST) symbols.
;  2943     3066  5              !
;  2944     3067  5              RST$SET_SCOPE = TRUE;
;  2945     3068  5              DBG$STA_GETSYMBOL(.PATHNAME, RSTPTR, KIND, 0, 0, 0, 0);
;  2946     3069  5              IF .RSTPTR NEQ 0
;  2947     3070  5              THEN
;  2948     3071  5                  IF .RSTPTR[RST$V_GLOBAL] THEN KIND = RST$K_INVALID;
;  2949     3072  5
;  2950     3073  5              CASE .KIND FROM RST$K_KIND_MINIMUM TO RST$K_KIND_MAXIMUM OF
;  2951     3074  5                  SET
;  2952     3075  5
;  2953     3076  5
;  2954     3077  5                  ! If the symbol is not unique, return with an error.  If
;  2955     3078  5                  ! this is the second time through, cancel the module we SET
;  2956     3079  5                  ! the last time through; then signal not unique.
;  2957     3080  5                  !
;  2958     3081  5                  [RST$K_NOTUNIQUE,
;  2959     3082  5                   RST$K_OVERLOAD]:
;  2960     3083  6                      BEGIN
;  2961     3084  6                      IF .MODPTR NEQ 0
;  2962     3085  6                      THEN
;  2963     3086  6                          STATUS = DBG$RST_CANMOD(PNAME[1], .PNAME[0]);
;  2964     3087  6
;  2965     3088  6                      IF .KIND EQL RST$K_NOTUNIQUE
;  2966     3089  6                      THEN
;  2967     3090  6                          SIGNAL(DBG$_NOUNIQUE, 1, .PATH_STRING)
;  2968     3091  6
;  2969     3092  6                      ELSE
;  2970     3093  6                          SIGNAL(DBG$_NOTUNQOVR, 1, .PATH_STRING);
;  2971     3094  6
;  2972     3095  5                      END;
;  2973     3096  5
;  2974     3097  5
;  2975     3098  5                  ! If the symbol is a lexical entity, it is a valid scope.
;  2976     3099  5                  ! Clear MODPTR to indicate this.
;  2977     3100  5                  !
;  2978     3101  5                  [RST$K_ROUTINE, RST$K_BLOCK]:
;  2979     3102  5                      MODPTR = 0;
;  2980     3103  5
;  2981     3104  5
;  2982     3105  5                  ! If the symbol is a Line Number, the actual scope is the
;  2983     3106  5                  ! containing lexical entity.  Change RSTPTR accordingly and
;  2984     3107  5                  ! set MODPTR to zero to indicate we have a good scope.
;  2985     3108  5                  !
;  2986     3109  5                  [RST$K_LINE]:
;  2987     3110  6                      BEGIN
;  2988     3111  6                      RSTPTR = .RSTPTR[RST$L_UPSCOPEPTR];
;  2989     3112  6                      KIND = .RSTPTR[RST$B_KIND];
;  2990     3113  6                      MODPTR = 0;
;  2991     3114  5                      END;
```

```
;  2992     3115  5
;  2993     3116  5
;  2994     3117  5                    ! If the symbol is anything else, this is not a valid scope.
;  2995     3118  5                    ! We thus try to interpret the first name in the pathname as
;  2996     3119  5                    ! a module name.  If this succeeds, we SET that module and
;  2997     3120  5                    ! loop to look up the scope again.
;  2998     3121  5                    !
;  2999     3122  5                    [INRANGE]:
;  3000     3123  6                        BEGIN
;  3001     3124  6
;  3002     3125  6
;  3003     3126  6                        ! See if this is the second time around (i.e.,
;  3004     3127  6                        ! we tried setting the module last time around).
;  3005     3128  6                        !
;  3006     3129  6                        IF .MODPTR NEQ 0
;  3007     3130  6                        THEN
;  3008     3131  7                            BEGIN
;  3009     3132  7
;  3010     3133  7                            ! If there was only one name in the pathname then
;  3011     3134  7                            ! what has happened is that we set the module
;  3012     3135  7                            ! and looped around to see if we found a scope
;  3013     3136  7                            ! within the module that we should use. We did
;  3014     3137  7                            ! not, so we use the module as the scope.
;  3015     3138  7                            !
;  3016     3139  7                            IF .PATHNAME[PTH$B_TOTCNT] EQL 1
;  3017     3140  7                            THEN
;  3018     3141  8                                BEGIN
;  3019     3142  8                                RSTPTR = .MODPTR;
;  3020     3143  8                                EXITLOOP;
;  3021     3144  7                                END;
;  3022     3145  7
;  3023     3146  7                            ! Otherwise, cancel the module and return error.
;  3024     3147  7                            !
;  3025     3148  7                            STATUS = DBG$RST_CANMOD(PNAME[1], .PNAME[0]);
;  3026     3149  7                            SIGNAL(DBG$_BADSCOPE, 1, .PATH_STRING);
;  3027     3150  6                            END;
;  3028     3151  6
;  3029     3152  6
;  3030     3153  6                        ! First time around.  Try to SET the module given by the
;  3031     3154  6                        ! first name in the Pathname Descriptor.
;  3032     3155  6                        !
;  3033     3156  6                        RSTPTR = GET_MOD_RST(PNAME[1], .PNAME[0]);
;  3034     3157  6
;  3035     3158  6
;  3036     3159  6                        ! If the first name is not a module, signal an error.
;  3037     3160  6                        !
;  3038     3161  6                        IF .RSTPTR EQL 0 THEN SIGNAL(DBG$_BADSCOPE, 1, .PATH_STRING);
;  3039     3162  6
;  3040     3163  6
;  3041     3164  6                        ! If the module is already set, and the pathname is more
;  3042     3165  6                        ! than a single name, then there is no use going on with
;  3043     3166  6                        ! the search.  Signal an error.
;  3044     3167  6                        !
;  3045     3168  6                        IF .RSTPTR[RST$V_MODSET] AND
;  3046     3169  7                           (.PATHNAME[PTH$B_TOTCNT] NEQ 1)
;  3047     3170  6                        THEN
;  3048     3171  6                            SIGNAL(DBG$_BADSCOPE, 1, .PATH_STRING);
```

```
;  3049          3172  6
;  3050          3173  6
;  3051          3174  6                    ! If the module is not set, try to set it.  Only if
;  3052          3175  6                    ! we are called from DBG$RST_SETSCOPE.
;  3053          3176  6                    !
;  3054          3177  6                    IF NOT .RSTPTR[RST$V_MODSET]
;  3055          3178  6                    THEN
;  3056          3179  7                        BEGIN
;  3057          3180  7                        IF NOT .ORIGIN_FLAG
;  3058          3181  7                        THEN
;  3059          3182  7                            SIGNAL(DBG$_BADSCOPE, 1, .PATH_STRING);
;  3060          3183  7                        IF NOT DBG$RST_SETMOD(PNAME[1], .PNAME[0], FALSE)
;  3061          3184  7                        THEN
;  3062          3185  7                            $DBG_ERROR('RSTCNTRL\RST_BLDSCOPE_LIST');
;  3063          3186  6                        END;
;  3064          3187  6
;  3065          3188  6
;  3066          3189  6                    ! Set MODPTR to point to the module we have just
;  3067          3190  6                    ! set and loop back to try looking up the name
;  3068          3191  6                    ! in this module.
;  3069          3192  6                    !
;  3070          3193  6                    MODPTR = .RSTPTR;
;  3071          3194  5                    END;
;  3072          3195  5
;  3073          3196  5                TES;
;  3074          3197  5
;  3075          3198  5
;  3076          3199  5            ! If MODPTR is non-zero, we have a new module to try the look-up
;  3077          3200  5            ! in, so we loop.  Otherwise, we exit this WHILE loop.
;  3078          3201  5            !
;  3079          3202  5            IF .MODPTR EQL 0 THEN EXITLOOP;
;  3080          3203  4            END;
;  3081          3204  4
;  3082          3205  4
;  3083          3206  4            ! We have now found the scope in the RST.  Create a Normal scope
;  3084          3207  4            ! entry for it.
;  3085          3208  4            !
;  3086          3209  4            MODPTR = .RSTPTR;
;  3087          3210  4            WHILE .MODPTR[RST$B_KIND] NEQ RST$K_MODULE DO
;  3088          3211  4                MODPTR = .MODPTR[RST$L_UPSCOPEPTR];
;  3089          3212  4
;  3090          3213  4            SCOPE = DBG$GET_MEMORY(SCOPE$K_ENTSIZE);
;  3091          3214  4            SCOPE[SCOPE$L_FLINK] = 0;
;  3092          3215  4            SCOPE[SCOPE$L_STATE] = SCOPE$K_NORMAL;
;  3093          3216  4            SCOPE[SCOPE$L_RSTPTR] = .RSTPTR;
;  3094          3217  4            SCOPE[SCOPE$L_MODPTR] = .MODPTR;
;  3095          3218  4
;  3096          3219  3            END;                        ! End of normal sccpe ELSE clause
;  3097          3220  3
;  3098          3221  3
;  3099          3222  3        ! We have now created a Scope Entry of some kind.  Attach it to the
;  3100          3223  3        ! end of the new scope list we are building.
;  3101          3224  3        !
;  3102          3225  3        PREVSCOPE[SCOPE$L_FLINK] = .SCOPE;
;  3103          3226  3        PREVSCOPE = .SCOPE;
;  3104          3227  3
;  3105          3228  2        END;                            ! End of INCR loop over PNVEC vector
```

```
;  3106          3229  2
;  3107          3230  2
;  3108          3231  2      ! We have now built the new scope list successfully.  Note this by setting
;  3109          3232  2      ! ERRINDEX to zero.
;  3110          3233  2      !
;  3111          3234  2      ERRINDEX[0] = 0;
;  3112          3235  2      RETURN .NEW_SCOPE_LIST;
;  3113          3236  2
;  3114          3237  1      END;


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

42  5F  54  53  52  5C  4C  52  54  4E  43  54  53  52  1A  000CA P.AAM:  .ASCII  <26>\RSTCNTRL\<92>\RST_BLDSCOPE_LIST\
                54  53  49  4C  5F  45  50  4F  43  53  44  4C  000D9
42  5F  54  53  52  5C  4C  52  54  4E  43  54  53  52  1A  000E5 P.AAN:  .ASCII  <26>\RSTCNTRL\<92>\RST_BLDSCOPE_LIST\
                54  53  49  4C  5F  45  50  4F  43  53  44  4C  000F4


                                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                        OFFC 00000              .ENTRY  DBG$RST_BLDSCOPE_LIST, Save R2,R3,R4,R5,R6,-  ;  2830
                                                                        R7,R8,R9,R10,R11
                                5B 00000000G   00  9E 00002     MOVAB   LIB$SIGNAL, R11
                                5E             10  C2 00009     SUBL2   #16, SP
                                          0C   AE  D4 0000C     CLRL    NEW_SCOPE_LIST                                  ;  2875
                                6D      02C8   CF  DE 0000F     MOVAL   42$, (FP)
                                58        0C   AE  9E 00014     MOVAB   NEW_SCOPE_LIST, PREVSCOPE                       ;  2905
                                               68  D4 00018     CLRL    (PREVSCOPE)                                     ;  2906
                                          04   BC  D5 0001A     TSTL    @PNVEC                                          ;  2912
                                               1B  12 0001D     BNEQ    1$
                                               04  DD 0001F     PUSHL   #4                                              ;  2915
                00000000G      00              01  FB 00021     CALLS   #1, DBG$GET_MEMORY
                               52              50  D0 00028     MOVL    R0, SCOPE
                                               62  D4 0002B     CLRL    (SCOPE)                                         ;  2916
                            04 A2              02  7D 0002D     MOVQ    #2, 4(SCOPE)                                    ;  2917
                                          0C   A2  D4 00031     CLRL    12(SCOPE)                                       ;  2919
                               68              52  D0 00034     MOVL    SCOPE, (PREVSCOPE)                              ;  2920
                               58              52  D0 00037     MOVL    SCOPE, PREVSCOPE                                ;  2921
                                               57  D4 0003A 1$:  CLRL    I                                               ;  2952
                                         028D  31  3C 0003C     BRW     41$
                            08 BC              57  D0 0003F 2$:  MOVL    I, @ERRINDEX                                    ;  2930
                               56        04 BC47  D0 00043     MOVL    @PNVEC[I], PATHNAME                             ;  2937
                               59        08   A6  9E 00048     MOVAB   8(R6), PATHVEC                                  ;  2938
                               53              69  D0 0004C     MOVL    (PATHVEC), PNAME                                ;  2939
                                       4040    8F  BB 0004F     PUSHR   #^M<R6,SP>                                      ;  2940
                00000000G      00              02  FB 00053     CALLS   #2, DBG$NPATHDESC_TO_CS
                            01 A6              66  91 0005A     CMPB    (PATHNAME), 1(PATHNAME)                         ;  2945
                                               0D  13 0005E     BEQL    3$
                                               6E  DD 00060     PUSHL   PATH_STRING                                     ;  2947
                                               01  DD 00062     PUSHL   #1
                                  000281D8     8F  DD 00064     PUSHL   #164312
                               6B              03  FB 0006A     CALLS   #3, LIB$SIGNAL
                               03        10    AC  E8 0006D 3$:  BLBS    MODULE_FLAG, 4$                                 ;  2961
                                         0093  31  00071        BRW     11$
```

```
                01        01    A6  91 00074 4$:     CMPB    1(PATHNAME), #1                 ;2958
                                0D  13 00078          BEQL    5$
                                6E  DD 0007A          PUSHL   PATH_STRING                     ;2960
                                01  DD 0007C          PUSHL   #1
                    00028100    8F  DD 0007E          PUSHL   #164096
                6B              03  FB 00084          CALLS   #3, LIB$SIGNAL
                                63  95 00087 5$:      TSTB    (PNAME)                         ;2961
                                0D  12 00089          BNEQ    6$
                                6E  DD 0008B          PUSHL   PATH_STRING                     ;2963
                                01  DD 0008D          PUSHL   #1
                    00028100    8F  DD 0008F          PUSHL   #164096
                6B              03  FB 00095          CALLS   #3, LIB$SIGNAL
                7E              63  9A 00098 6$:      MOVZBL  (PNAME), -(SP)                  ;2968
                        01      A3  9F 0009B          PUSHAB  1(PNAME)
        0000V   CF              02  FB 0009E          CALLS   #2, GET_MOD_RST
          08    AE              50  D0 000A3          MOVL    R0, RSTPTR
                55        08    AE  D0 000A7          MOVL    RSTPTR, R5                      ;2973
                                0D  12 000AB          BNEQ    7$
                                6E  DD 000AD          PUSHL   PATH_STRING
                                01  DD 000AF          PUSHL   #1
                    000281E8    8F  DD 000B1          PUSHL   #164328
                6B              03  FB 000B7          CALLS   #3, LIB$SIGNAL
                21        28    A5  E8 000BA 7$:      BLBS    40(R5), 8$                      ;2978
                                7E  D4 000BE          CLRL    -(SP)                           ;2980
                7E              63  9A 000C0          MOVZBL  (PNAME), -(SP)
                        01      A3  9F 000C3          PUSHAB  1(PNAME)
        0000V   CF              03  FB 000C6          CALLS   #3, DBG$RST_SETMOD
                11              50  E8 000CB          BLBS    R0, 8$
                    00000000'   EF  9F 000CE          PUSHAB  P.AAM                           ;2982
                                01  DD 000D4          PUSHL   #1
                    00028362    8F  DD 000D6          PUSHL   #164706
                6B              03  FB 000DC          CALLS   #3, LIB$SIGNAL
                54              55  D0 000DF 8$:      MOVL    R5, MODPTR                      ;2988
                01        14    A4  91 000E2 9$:      CMPB    20(MODPTR), #1                  ;2989
                                06  13 000E6          BEQL    10$
                54        10    A4  D0 000E8          MOVL    16(MODPTR), MODPTR              ;2990
                                F4  11 000EC          BRB     9$
                                04  DD 000EE 10$:      PUSHL   #4                              ;2992
  00000000G     00              01  FB 000F0          CALLS   #1, DBG$GET_MEMORY
                52              50  D0 000F7          MOVL    R0, SCOPE
                                62  D4 000FA          CLRL    (SCOPE)                         ;2993
          04    A2              01  D0 000FC          MOVL    #1, 4(SCOPE)                    ;2994
          08    A2              55  D0 00100          MOVL    R5, 8(SCOPE)                    ;2995
                            01BB  31 00104          BRW     39$                             ;2996
                                63  95 00107 11$:     TSTB    (PNAME)                         ;3003
                                5D  12 00109          BNEQ    16$
                01              66  91 0010B          CMPB    (PATHNAME), #1                  ;3006
                                0D  13 0010E          BEQL    12$
                                6E  DD 00110          PUSHL   PATH_STRING                     ;3008
                                01  DD 00112          PUSHL   #1
                    000281D8    8F  DD 00114          PUSHL   #164312
                6B              03  FB 0011A          CALLS   #3, LIB$SIGNAL
                        02      A6  95 0011D 12$:     TSTB    2(PATHNAME)                     ;3014
                                17  12 00120          BNEQ    13$
                                04  DD 00122          PUSHL   #4                              ;3017
  00000000G     00              01  FB 00124          CALLS   #1, DBG$GET_MEMORY
                52              50  D0 0012B          MOVL    R0, SCOPE
```

```
                                 62  D4 0012E          CLRL    (SCOPE)                                 ; 3018
                              04 A2  03  7D 00130          MOVQ    #3, 4(SCOPE)                            ; 3019
                           0C    A2  2C  D4 00134          CLRL    12(SCOPE)                               ; 3021
                                     2C  11 00137          BRB     15$                                     ; 3014
                           01        02  A6 91 00139 13$:  CMPB    2(PATHNAME), #1                         ; 3028
                                     19  12 0013D          BNEQ    14$
                                     04  DD 0013F          PUSHL   #4                                      ; 3031
                00000000G    00        01  FB 00141          CALLS   #1, DBG$GET_MEMORY
                             52        50  D0 00148          MOVL    R0, SCOPE
                                       62  D4 0014B          CLRL    (SCOPE)                                 ; 3032
                              04 A2    02  7D 0014D          MOVQ    #2, 4(SCOPE)                            ; 3033
                           0C    A2  04  A6  D0 00151          MOVL    4(PATHNAME), 12(SCOPE)                  ; 3035
                                     0D  11 00156          BRB     15$                                     ; 3028
                                     6E  DD 00158 14$:     PUSHL   PATH_STRING                             ; 3042
                                     01  DD 0015A          PUSHL   #1
                           000281D8      8F  DD 0015C          PUSHL   #164312
                             6B          03  FB 00162          CALLS   #3, LIB$SIGNAL
                                   015E  31  00165 15$:      BRW     40$                                     ; 3003
                                     54  D4 00168 16$:      CLRL    MODPTR                                  ; 3059
                00000000'    EF        01  D0 0016A 17$:     MOVL    #1, RST$SET_SCOPE                       ; 3067
                                       7E  7C 00171          CLRQ    -(SP)                                   ; 3068
                                       7E  7C 00173          CLRQ    -(SP)
                                 14    AE  9F 00175          PUSHAB  KIND
                                 1C    AE  9F 00178          PUSHAB  RSTPTR
                                       56  DD 0017B          PUSHL   PATHNAME
                00000000G    00        07  FB 0017D          CALLS   #7, DBG$STA_GETSYMBOL
                             50        08  AE  D0 00184          MOVL    RSTPTR, R0                      ; 3069
                                       07  13 00188          BEQL    18$
                             03        15  A0  E9 0018A          BLBC    21(R0), 18$                     ; 3071
                                 04    AE  D4 0018E          CLRL    KIND
                          0D           00  04  AE  CF 00191 18$:  CASEL   KIND, #0, #13                 ; 3073
                005D      005D           0062      0062  00196 19$:  .WORD   27$-19$,-
                0062      0062           004F      0062  0019E          27$-19$,-
                0062      0062           001C      0062  001A6          25$-19$,-
                                         001C      0062  001AE          25$-19$,-
                                                                      27$-19$,-
                                                                      24$-19$,-
                                                                      27$-19$,-
                                                                      27$-19$,-
                                                                      27$-19$,-
                                                                      20$-19$,-
                                                                      27$-19$,-
                                                                      27$-19$,-
                                                                      27$-19$,-
                                                                      20$-19$
                                       54  D5 001B2 20$:      TSTL    MODPTR                          ; 3084
                                       0E  13 001B4          BFQL    21$
                                     7E        63  9A 001B6          MOVZBL  (PNAME), -(SP)          ; 3086
                                 01    A3  9F 001B9          PUSHAB  1(PNAME)
                          0000V  CF        02  FB 001BC          CALLS   #2, DBG$RST_CANMOD
                                 5A        50  D0 001C1          MOVL    R0, STATUS
                                 09  04    AE  D1 001C4 21$:     CMPL    KIND, #9                        ; 3088
                                           0C  12 001C8          BNEQ    22$
                                           6E  DD 001CA          PUSHL   PATH_STRING                     ; 3090
                                           01  DD 001CC          PUSHL   #1
                                 000281F0      8F  DD 001CE          PUSHL   #164336
                                           0A  11 001D4          BRB     23$
```

```
                        6E  DD 001D6 22$:  PUSHL   PATH_STRING                      ; 3093
                        01  DD 001D8       PUSHL   #1                              ;
              000282A8  8F  DD 001DA       PUSHL   #164520                         ;
        6B              03  FB 001E0 23$:  CALLS   #3, LIB$SIGNAL                  ;
                        10  11 001E3       BRB     26$                             ; 3073
  08    AE        10    A0  D0 001E5 24$:  MOVL    16(R0), RSTPTR                  ; 3111
        50        08    AE  D0 001EA       MOVL    RSTPTR, R0                      ; 3112
  04    AE        14    A0  9A 001EE       MOVZBL  20(R0), KIND                    ;
                        54  D4 001F3 25$:  CLRL    MODPTR                          ; 3113
                    009C  31 001F5 26$:  BRW     35$                             ; 3073
                        54  D5 001F8 27$:  TSTL    MODPTR                          ; 3129
                        27  13 001FA       BEQL    29$                             ;
        01              66  91 001FC       CMPB    (PATHNAME), #1                  ; 3139
                        07  12 001FF       BNEQ    28$                             ;
  08    AE              54  D0 00201       MOVL    MODPTR, RSTPTR                  ; 3142
                    0093  31 00205       BRW     36$                             ; 3141
        7E              63  9A 00208 28$:  MOVZBL  (PNAME), -(SP)                  ; 3148
                  01    A3  9F 0020B       PUSHAB  1(PNAME)                        ;
0000V   CF              02  FB 0020E       CALLS   #2, DBG$RST_CANMOD              ;
        5A              50  D0 00213       MOVL    R0, STATUS                      ;
                        6E  DD 00216       PUSHL   PATH_STRING                     ; 3149
                        01  DD 00218       PUSHL   #1                              ;
              000281D8  8F  DD 0021A       PUSHL   #164312                         ;
        6B              03  FB 00220       CALLS   #3, LIB$SIGNAL                  ;
        7E              63  9A 00223 29$:  MOVZBL  (PNAME), -(SP)                  ; 3156
                  01    A3  9F 00226       PUSHAB  1(PNAME)                        ;
0000V   CF              02  FB 00229       CALLS   #2, GET_MOD_RST                 ;
  08    AE              50  D0 0022F       MOVL    R0, RSTPTR                      ;
        55        08    AE  D0 00232       MOVL    RSTPTR, R5                      ; 3161
                        0D  12 00236       BNEQ    30$                             ;
                        6E  DD 00238       PUSHL   PATH_STRING                     ;
                        01  DD 0023A       PUSHL   #1                              ;
              000281D8  8F  DD 0023C       PUSHL   #164312                         ;
        6B              03  FB 00242       CALLS   #3, LIB$SIGNAL                  ;
        16        28    A5  E9 00245 30$:  BLBC    40(R5), 32$                     ; 3168
        01              66  91 00249       CMPB    (PATHNAME), #1                  ; 3169
                        0D  13 0024C       BEQL    31$                             ;
                        6E  DD 0024E       PUSHL   PATH_STRING                     ; 3171
                        01  DD 00250       PUSHL   #1                              ;
              000281D8  8F  DD 00252       PUSHL   #164312                         ;
        6B              03  FB 00258       CALLS   #3, LIB$SIGNAL                  ;
        32        28    A5  E8 0025B 31$:  BLBS    40(R5), 34$                     ; 3177
        0D        0C    AC  E8 0025F 32$:  BLBS    ORIGIN_FLAG, 33$                ; 3180
                        6E  DD 00263       PUSHL   PATH_STRING                     ; 3182
                        01  DD 00265       PUSHL   #1                              ;
              000281D8  8F  DD 00267       PUSHL   #164312                         ;
        6B              03  FB 0026D       CALLS   #3, LIB$SIGNAL                  ;
        7E              D4  00270 33$:  CLRL    -(SP)                           ; 3183
        7E              63  9A 00272       MOVZBL  (PNAME), -(SP)                  ;
                  01    A3  9F 00275       PUSHAB  1(PNAME)                        ;
  0000V CF              03  FB 00278       CALLS   #3, DBG$RST_SETMOD              ;
        11              50  E8 0027D       BLBS    R0, 34$                         ;
              00000000'  EF  9F 00280       PUSHAB  P.AAN                           ; 3185
                        01  DD 00286       PUSHL   #1                              ;
              00028362  8F  DD 00288       PUSHL   #164706                         ;
        6B              03  FB 0028E       CALLS   #3, LIB$SIGNAL                  ;
        54              55  D0 00291 34$:  MOVL    R5, MODPTR                      ; 3193
```

```
                                    54  D5 00294 35$:   TSTL    MODPTR                                  ; 3202
                                    03  13 00296       BEQL    36$                                     ;
                                FECF    31 00298       BRW     17$                                     ;
                          54    08  AE  D0 0029B 36$:   MOVL    RSTPTR, MODPTR                          ; 3209
                          01    14  A4  91 0029F 37$:   CMPB    20(MODPTR), #1                          ; 3210
                                    06  13 002A3       BEQL    38$                                     ;
                          54    10  A4  D0 002A5       MOVL    16(MODPTR), MODPTR                      ; 3211
                                    F4  11 002A9       BRB     37$                                     ;
                                    04  DD 002AB 38$:   PUSHL   #4                                      ; 3213
                00000000G   00          01  FB 002AD       CALLS   #1, DBG$GET_MEMORY                  ;
                            52          50  D0 002B4       MOVL    R0, SCOPE                           ;
                                        62  D4 002B7       CLRL    (SCOPE)                             ; 3214
                      04    A2          01  D0 002B9       MOVL    #1, 4(SCOPE)                        ; 3215
                      08    A2    08    AE  D0 002BD       MOVL    RSTPTR, 8(SCOPE)                    ; 3216
                      0C    A2          54  D0 002C2 39$:   MOVL    MODPTR, 12(SCOPE)                   ; 3217
                            68          52  D0 002C6 40$:   MOVL    SCOPE, (PREVSCOPE)                  ; 3225
                            58          52  D0 002C9       MOVL    SCOPE, PREVSCOPE                    ; 3226
          FD6C        57            01    04  BC  F1 002CC 41$:   ACBL    @PNVEC, #1, I, 2$           ; 2928
                                      08    BC  D4 002D3       CLRL    @ERRINDEX                       ; 3234
                            50        0C    AE  D0 002D6       MOVL    NEW_SCOPE_LIST, R0              ; 3235
                                            04  002DA       RET                                     ; 3237
                                          0000  002DB 42$:   .WORD   Save nothing                    ; 2875
                            50        08    AC  D0 002DD       MOVL    8(AP), R0                       ;
                            50        04    A0  D0 002E1       MOVL    4(R0), R0                       ;
                                      FC    A0  9F 002E5       PUSHAB  NEW_SCOPE_LIST                  ;
                                            01  DD 002E8       PUSHL   #1                              ;
                                            5E  DD 002EA       PUSHL   SP                              ;
                            7E        04    AC  7D 002EC       MOVQ    4(AP), -(SP)                    ;
                      0000V CF              03  FB 002F0       CALLS   #3, BLDSCOPE_ERROR_HANDLER      ;
                                            04  002F5       RET                                     ;

; Routine Size:  758 bytes,    Routine Base:  DBG$CODE + 17D1
```

```
;3116     3238  1 GLOBAL ROUTINE DBG$RST_CANMOD(MODNAMEPTR, LENGTH) =
;3117     3239  1 !
;3118     3240  1 ! FUNCTION
;3119     3241  1 !       This routine marks a specified module as being CANCELLED (not SET).
;3120     3242  1 !       The routine accepts the desired module name and its length as input,
;3121     3243  1 !       looks up that name in the RST Module Chain, and marks the module as
;3122     3244  1 !       being not SET.  Alternatively, if no module name is specified, all
;3123     3245  1 !       modules in the RST Module Chain are CANCELLED.
;3124     3246  1 !
;3125     3247  1 ! INPUTS
;3126     3248  1 !       MODNAMEPTR - A pointer to the first character of an ASCII string
;3127     3249  1 !                    containing the module name.  If this pointer is zero, it
;3128     3250  1 !                    is assumed that all modules should be marked as not SET.
;3129     3251  1 !
;3130     3252  1 !       LENGTH  - The length in characters of the module name.
;3131     3253  1 !
;3132     3254  1 ! OUTPUTS
;3133     3255  1 !       This routine returns the value STS$K_SUCCESS if the module was success-
;3134     3256  1 !                    fully cancelled, or it returns STS$K_SEVERE if the specified
;3135     3257  1 !                    module name does not exist in the RST Module Chain.  If all
;3136     3258  1 !                    modules are cancelled, STS$K_SUCCESS is returned.
;3137     3259  1 !
;3138     3260  1
;3139     3261  2     BEGIN
;3140     3262  2
;3141     3263  2     LOCAL
;3142     3264  2         RSTPTR: REF RST$ENTRY;          ! Pointer to the Module RST Entry
;3143     3265  2
;3144     3266  2
;3145     3267  2
;3146     3268  2     ! If a specific module name was specified, we look up the corresponding
;3147     3269  2     ! Module RST Entry, mark the module as being CANCELLED (not SET), and remove
;3148     3270  2     ! the module's RST symbol chain.
;3149     3271  2     !
;3150     3272  2     IF .MODNAMEPTR NEQ 0
;3151     3273  2     THEN
;3152     3274  3         BEGIN
;3153     3275  3         RSTPTR = GET_MOD_RST(.MODNAMEPTR, .LENGTH);
;3154     3276  3         IF .RSTPTR EQL 0 THEN RETURN STS$K_SEVERE;
;3155     3277  3         RSTPTR[RST$V_MODSET] = FALSE;
;3156     3278  3         DBG$RST_REMOVE(.RSTPTR);
;3157     3279  3         RETURN STS$K_SUCCESS;
;3158     3280  2         END;
;3159     3281  2
;3160     3282  2
;3161     3283  2     ! No module name was specified.  Go through the whole RST Module Chain (but
;3162     3284  2     ! ignore the anonymous module) and mark each module as being not SET.  Also
;3163     3285  2     ! remove each such module's RST symbol chain.  Note that we abort this loop
;3164     3286  2     ! if the user has entered Control-Y DEBUG to stop the command.
;3165     3287  2     !
;3166     3288  2     RSTPTR = .RST$START_ADDR;
;3167     3289  2     WHILE .RSTPTR NEQ 0 DO
;3168     3290  3         BEGIN
;3169     3291  3         $ABORT_ON_CONTROL_Y;
;3170     3292  3         IF NOT .RSTPTR[RST$V_ANONMOD]
;3171     3293  3         THEN
;3172     3294  4             BEGIN
```

```
;  3173          3295  4              RSTPTR[RST$V_MODSET] = FALSE;
;  3174          3296  4              DBG$RST_REMOVE(.RSTPTR);
;  3175          3297  3              END;
;  3176          3298  3
;  3177          3299  3          RSTPTR = .RSTPTR[RST$L_NXTMODPTR];
;  3178          3300  2          END;
;  3179          3301  2
;  3180          3302  2      RETURN STS$K_SUCCESS;
;  3181          3303  2
;  3182          3304  1      END;


                                                0004 00000          .ENTRY  DBG$RST_CANMOD, Save R2                 ; 3238
                                        04      AC   D5 00002       TSTL    MODNAMEPTR                              ; 3272
                                                1F   13 00005       BEQL    2$
                                    7E          04   AC  7D 00007   MOVQ    MODNAMEPTR, -(SP)                       ; 3275
                                0000V   CF              02   FB 0000B   CALLS   #2, GET_MOD_RST
                                        52              50   D0 00010   MOVL    R0, RSTPTR
                                                        04   12 00013   BNEQ    1$                                  ; 3276
                                        50              04   D0 00015   MOVL    #4, R0
                                                        04   00018     RET
                                  28    A2              01   8A 00019 1$:   BICB2   #1, 40(RSTPTR)                  ; 3277
                                                        52   DD 0001D   PUSHL   RSTPTR                              ; 3278
                                0000V   CF              01   FB 0001F   CALLS   #1, DBG$RST_REMOVE
                                                        34   11 00024   BRB     6$                                  ; 3279
                                        52 00000000'    EF   D0 00026 2$:   MOVL    RST$START_ADDR, RSTPTR          ; 3288
                                                        2B   13 0002D 3$:   BEQL    6$                              ; 3289
                            0D 00000000G    00          01   E1 0002F   BBC     #1, DBG$GV_CONTROL+1, 4$            ; 3290
                                            000280E8    8F   DD 00037   PUSHL   #164072
                                00000000G   00          01   FB 0003D   CALLS   #1, LIB$SIGNAL
                            08      28      A2          02   E0 00044 4$:   BBS     #2, 40(RSTPTR), 5$              ; 3292
                                    28      A2          01   8A 00049   BICB2   #1, 40(RSTPTR)                      ; 3295
                                                        52   DD 0004D   PUSHL   RSTPTR                              ; 3296
                                    0000V   CF          01   FB 0004F   CALLS   #1, DBG$RST_REMOVE
                                            52      10  A2   D0 00054 5$:   MOVL    16(RSTPTR), RSTPTR              ; 3299
                                                        D3   11 00058   BRB     3$                                  ; 3289
                                            50          01   D0 0005A 6$:   MOVL    #1, R0                          ; 3302
                                                        04   0005D     RET                                         ; 3304

; Routine Size:  94 bytes,    Routine Base:  DBG$CODE + 1AC7
```

```
;  3184          3305  1 GLOBAL ROUTINE DBG$RST_DST_LENGTH(PASSED_DSTPTR) =
;  3185          3306  1 !
;  3186          3307  1 ! FUNCTION
;  3187          3308  1 !       This routine searches the Moved DST entry list for a matching
;  3188          3309  1 !       "real" DST pointer.  If it finds one it returns the length of the
;  3189          3310  1 !       DST as stored in the MDE$ structure.  If not it returns the DST length
;  3190          3311  1 !       from the record.
;  3191          3312  1 !
;  3192          3313  1 ! INPUTS
;  3193          3314  1 !       PASSED_DSTPTR - The DST pointer to look for. Passed by value.
;  3194          3315  1 !
;  3195          3316  1 ! OUTPUTS
;  3196          3317  1 !       Returns the DST length.
;  3197          3318  1 !
;  3198          3319  1
;  3199          3320  2     BEGIN
;  3200          3321  2
;  3201          3322  2     LOCAL
;  3202          3323  2         DSTPTR: REF DST$RECORD,          ! ???
;  3203          3324  2         MOVED_DST_ENTRY: REF MDE$RECORD;! ???
;  3204          3325  2
;  3205          3326  2
;  3206          3327  2
;  3207          3328  2     ! Assure ourselves that it is "real".
;  3208          3329  2     !
;  3209          3330  2     DSTPTR = DBG$RST_DST_PTR(.PASSED_DSTPTR);
;  3210          3331  2     MOVED_DST_ENTRY = .DBG$GL_MOVED_DST_LIST_HEAD;
;  3211          3332  2     WHILE .MOVED_DST_ENTRY NEQA 0 DO
;  3212          3333  3         BEGIN
;  3213          3334  3
;  3214          3335  3
;  3215          3336  3         ! Check if we've found it and return the "next" one.
;  3216          3337  3         !
;  3217          3338  3         IF .DSTPTR EQLA .MOVED_DST_ENTRY[MDE$L_REAL_DSTPTR]
;  3218          3339  3         THEN
;  3219          3340  3             RETURN .MOVED_DST_ENTRY[MDE$L_REAL_LENGTH];
;  3220          3341  3
;  3221          3342  3
;  3222          3343  3         ! Next!
;  3223          3344  3         !
;  3224          3345  3         MOVED_DST_ENTRY = .MOVED_DST_ENTRY[MDE$L_FLINK];
;  3225          3346  2         END;
;  3226          3347  2
;  3227          3348  2
;  3228          3349  2     ! We didn't find anything.
;  3229          3350  2     !
;  3230          3351  2     RETURN .DSTPTR[DST$B_LENGTH];
;  3231          3352  2
;  3232          3353  1     END;


                                0000 00000          .ENTRY  DBG$RST_DST_LENGTH, Save nothing        ; 3305
                          04    AC  DD 00002          PUSHL   PASSED_DSTPTR                           ; 3330
                    0000V CF          01  FB 00005          CALLS   #1, DBG$RST_DST_PTR             ;
```

```
                    51 00000000G   00  D0 0000A          MOVL    DBG$GL_MOVED_DST_LIST_HEAD, MOVED_DST_ENTRY ; 3331
                                   10  13 00011 1$:      BEQL    3$                                          ; 3332
                08  A1             50  D1 00013          CMPL    DSTPTR, 8(MOVED_DST_ENTRY)                  ; 3338
                                   05  12 00017          BNEQ    2$                                          ;
                    50        0C   A1  D0 00019          MOVL    12(MOVED_DST_ENTRY), R0                     ; 3340
                                       04 0001D          RET                                                 ;
                    51             61  D0 0001E 2$:      MOVL    (MOVED_DST_ENTRY), MOVED_DST_ENTRY          ; 3345
                                   EE  11 00021          BRB     1$                                          ; 3332
                    50             60  9A 00023 3$:      MOVZBL  (DSTPTR), R0                                ; 3351
                                       04 00026          RET                                                 ; 3353

; Routine Size:  39 bytes,    Routine Base:  DBG$CODE + 1B25
```

```
;  3234     3354  1 GLOBAL ROUTINE DBG$RST_DST_PTR(PASSED_DSTPTR) =
;  3235     3355  1 !
;  3236     3356  1 ! FUNCTION
;  3237     3357  1 !       This routine searches the Moved DST entry list for a matching
;  3238     3358  1 !       old DST pointer.  If it find one it returns the address of the
;  3239     3359  1 !       moved DST record.  If not it returns the passed DST pointer.
;  3240     3360  1 !
;  3241     3361  1 ! INPUTS
;  3242     3362  1 !       PASSED_DSTPTR - The DST pointer to look for.  Passed by value.
;  3243     3363  1 !
;  3244     3364  1 ! OUTPUTS
;  3245     3365  1 !       Returns the "real" DST pointer.
;  3246     3366  1 !
;  3247     3367  1
;  3248     3368  2     BEGIN
;  3249     3369  2
;  3250     3370  2     LOCAL
;  3251     3371  2         MOVED_DST_ENTRY: REF MDE$RECORD;! ???
;  3252     3372  2
;  3253     3373  2
;  3254     3374  2
;  3255     3375  2     ! ---
;  3256     3376  2     !
;  3257     3377  2     MOVED_DST_ENTRY = .DBG$GL_MOVED_DST_LIST_HEAD;
;  3258     3378  2     WHILE .MOVED_DST_ENTRY NEQ 0 DO
;  3259     3379  3         BEGIN
;  3260     3380  3
;  3261     3381  3
;  3262     3382  3         ! Check if we've found it and return the "real" one.
;  3263     3383  3         !
;  3264     3384  3         IF .PASSED_DSTPTR EQLA .MOVED_DST_ENTRY[MDE$L_ORIGINAL_DSTPTR]
;  3265     3385  3         THEN
;  3266     3386  3             RETURN .MOVED_DST_ENTRY[MDE$L_REAL_DSTPTR];
;  3267     3387  3
;  3268     3388  3
;  3269     3389  3         ! Next!
;  3270     3390  3         !
;  3271     3391  3         MOVED_DST_ENTRY = .MOVED_DST_ENTRY[MDE$L_FLINK];
;  3272     3392  3
;  3273     3393  2         END;
;  3274     3394  2
;  3275     3395  2
;  3276     3396  2     ! We didn't find anything.
;  3277     3397  2     !
;  3278     3398  2     RETURN .PASSED_DSTPTR;
;  3279     3399  2
;  3280     3400  1     END;


                                    0000 00000          .ENTRY  DBG$RST_DST_PTR, Save nothing                   ; 3354
                50 00000000G    00  D0 00002          MOVL    DBG$GL_MOVED_DST_LIST_HEAD, MOVED_DST_ENTRY     ; 3377
                                11  13 00009 1$:       BEQL    3$                                              ; 3378
          04    A0        04    AC  D1 0000B          CMPL    PASSED_DSTPTR, 4(MOVED_DST_ENTRY)               ; 3384
                                05  12 00010          BNEQ    2$                                              ;
```

```
                50      08    A0  D0 00012          MOVL    8(MOVED_DST_ENTRY), R0                      ; 3386
                                  04 00016          RET                                                 ;
                50            60  D0 00017 2$:      MOVL    (MOVED_DST_ENTRY), MOVED_DST_ENTRY          ; 3391
                              ED  11 0001A          BRB     1$                                          ; 3378
                50      04    AC  D0 0001C 3$:      MOVL    PASSED_DSTPTR, R0                           ; 3398
                                  04 00020          RET                                                 ; 3400

; Routine Size:  33 bytes,    Routine Base:  DBG$CODE + 1B4C
```

```
;  3282          3401  1 GLOBAL ROUTINE DBG$RST_INIT(IMGFILCHAN, FILENAME,
;  3283          3402  1                             IMG_HEADER, TRANSFER_ADDR): NOVALUE =
;  3284          3403  1 !
;  3285          3404  1 ! FUNCTION
;  3286          3405  1 !       This routine initializes the Debug Symbol Table (DST) and the Run-Time
;  3287          3406  1 !       Symbol Table (RST) when the Debugger is first invoked.  It locates the
;  3288          3407  1 !       DST in the user's executable image file and maps it into the Debugger's
;  3289          3408  1 !       (and thus the user's) virtual address space.  It does the same with the
;  3290          3409  1 !       Global Symbol Table (GST).  Thereafter, it scans the entire DST to find
;  3291          3410  1 !       all modules which have DST records.  For each such module, the amount of
;  3292          3411  1 !       RST space required for the module is recorded, the static address bounds
;  3293          3412  1 !       of the module are entered in the Program Static Address Table (SAT), and
;  3294          3413  1 !       a Module RST Entry is constructed.  The RST Hash Table is also initial-
;  3295          3414  1 !       ized and all module names are entered in it.  Finally, enough free memo-
;  3296          3415  1 !       ry space is allocated to accommodate the RSTs of the largest several
;  3297          3416  1 !       modules.  Thus it should always be possible to have several modules in
;  3298          3417  1 !       the RST at the same time no matter how large they are.
;  3299          3418  1 !
;  3300          3419  1 ! INPUTS
;  3301          3420  1 !       IMGFILCHAN - The channel that the image file is open on
;  3302          3421  1 !       FILENAME - The address of a Counted ASCII string containing the file-
;  3303          3422  1 !                  name of the executable image file.
;  3304          3423  1 !
;  3305          3424  1 !       IMG_HEADER - The address (in memory) of the executable image file's
;  3306          3425  1 !                  header block.
;  3307          3426  1 !
;  3308          3427  1 !       TRANSFER_ADDR - The program transfer address.  This is used only to
;  3309          3428  1 !                  select the module which is initially SET.
;  3310          3429  1 !
;  3311          3430  1 ! OUTPUTS
;  3312          3431  1 !       The DST, GST, and RST are initialized.  No value is returned.
;  3313          3432  1 !
;  3314          3433  1
;  3315          3434  2     BEGIN
;  3316          3435  2
;  3317          3436  2     MAP
;  3318          3437  2         FILENAME: REF VECTOR[,BYTE],     ! The .EXE file's name in counted ASCII
;  3319          3438  2         IMG_HEADER: REF BLOCK[,BYTE];    ! Pointer to the .EXE file image header
;  3320          3439  2
;  3321          3440  2     LITERAL
;  3322          3441  2         MAX_ROUT_STACK  = 50;            ! Maximum size of the internal routine
;  3323          3442  2                                          !      start address stack--deeper rout-
;  3324          3443  2                                          !      ine nesting can mess up SAT.
;  3325          3444  2
;  3326          3445  2     LOCAL
;  3327          3446  2         ADDRESS,                         ! Memory block address from $EXPREG
;  3328          3447  2         DSTBLK_COUNT,                    ! Number of DST blocks to map to memory
;  3329          3448  2         DSTPTR: REF DST$RECORD,          ! Pointer to the current DST record
;  3330          3449  2         ERROR_COUNT,                     ! Counts number of error messages given
;  3331          3450  2                                          !      for Missing Module-Begin Record
;  3332          3451  2         EXE_FILNAM_DESC: VECTOR[2,LONG], ! File name descriptor for $CREMAPSEC
;  3333          3452  2         EXE_SECBOUNDS: VECTOR[2,LONG],   ! Map section bounds descriptor for the
;  3334          3453  2                                          !      $CREMAPSEC system service
;  3335          3454  2         EXE_SECNAM_DESC: VECTOR[2,LONG], ! Map section name descriptor for the
;  3336          3455  2                                          !      $CREMAPSEC system service
;  3337          3456  2         IMG_DST_BLOCK: REF IHS$ENTRY,    ! Pointer to DST/GST descriptor in the
;  3338          3457  2                                          !      executable image header
```

```
;  3339     3458  2          IN_MODULE_FLAG,                 ! Set to TRUE if we are after a Module-
;  3340     3459  2                                          !      Begin record and before a Module-
;  3341     3460  2                                          !      End record in the DST
;  3342     3461  2          LANGUAGE,                       ! Language of the current module
;  3343     3462  2          LENGTH,                         ! Byte length of a static address range
;  3344     3463  2          LONGS_NEEDED,                   ! Number of longwords of RST space we
;  3345     3464  2                                          !      need from $EXPREG to handle
;  3346     3465  2                                          !      the largest several modules.
;  3347     3466  2          MOD_DSTPTR: REF DST$RECORD,     ! Pointer to the DST Module-Begin record
;  3348     3467  2                                          !      for the current module
;  3349     3468  2          MODNAMEPTR,                     ! Pointer to current module's name in
;  3350     3469  2                                          !      counted ASCII (used for errors).
;  3351     3470  2          MOD_SIZE_TBL:                   ! Table giving the RST symbol table
;  3352     3471  2                  VECTOR[MIN_MODS_IN_RST],!      sizes of the MIN_MODS_IN_RST
;  3353     3472  2                                          !      largest modules in longwords.
;  3354     3473  2          MST_INDEX,                      ! Index into MOD_SIZE_TBL vector
;  3355     3474  2          NEXTSAT,                        ! Pointer to the next SAT entry on the
;  3356     3475  2                                          !      TMP_SAT_CHAIN chain.
;  3357     3476  2          NUM_BYTES,                      ! Number of bytes in $EXPREG call
;  3358     3477  2          NUM_RST_ENTS,                   ! Number of RST entries needed for the
;  3359     3478  2                                          !      current module's symbol table
;  3360     3479  2          NUM_RST_LONGS,                  ! Number of RST longwords needed for the
;  3361     3480  2                                          !      current module's symbol table
;  3362     3481  2          NUM_SAT_ENTS,                   ! Number of Static Address Table entries
;  3363     3482  2                                          !      in this module's RST
;  3364     3483  2          PPTR: REF DST$PSECT_TRAILER,    ! Pointer to PSECT DST record trailer
;  3365     3484  2          PREV_SAT_PTR: REF SAT$ENTRY,    ! Pointer to previous SAT entry in the
;  3366     3485  2                                          !      Program SAT chain--used to insert
;  3367     3486  2                                          !      new Static Address Table entries.
;  3368     3487  2          PROG_SAT_PTR: REF SAT$ENTRY,    ! Pointer to current SAT entry in the
;  3369     3488  2                                          !      Program SAT chain--used to insert
;  3370     3489  2                                          !      new Static Address Table entries.
;  3371     3490  2          REC_NESTING,                    ! Depth of data record nesting in DST
;  3372     3491  2          ROUT_STACK:                     ! Routine start address stack--used to
;  3373     3492  2                  VECTOR[MAX_ROUT_STACK],!      build SAT for nested routines
;  3374     3493  2          ROUT_STACK_INDEX,               ! Current size of routine address stack
;  3375     3494  2          RSTPTR: REF RST$ENTRY,          ! Pointer to the current RST entry
;  3376     3495  2          RSTSIZE,                        ! The RST space in longwords needed for
;  3377     3496  2                                          !      the current module's symbol table
;  3378     3497  2          SATPTR: REF SAT$ENTRY,          ! Pointer to SAT entry when searching
;  3379     3498  2                                          !      for module with transfer address
;  3380     3499  2          STARTADDR,                      ! The start address of a static area
;  3381     3500  2          STATIC_FLAG,                    ! Flag set to TRUE for static data items
;  3382     3501  2          STATUS: BLOCK[1, LONG],         ! Status returned by RMS or $CREMAPSEC
;  3383     3502  2          TESTDSTSIZE,                    ! Size of test program DST for a
;  3384     3503  2                                          !      Testable Debugger
;  3385     3504  2          TMP_SAT_CHAIN,                  ! Pointer to a temporary Static Address
;  3386     3505  2                                          !      Table chain for current module
;  3387     3506  2          TMP_SAT_PTR: REF SAT$ENTRY,     ! Pointer to the current entry in the
;  3388     3507  2                                          !      temporary SAT chain when copying
;  3389     3508  2                                          !      to the Program SAT chain.
;  3390     3509  2          TYPE;                           ! The type of the current DST record
;  3391     3510  2
;  3392     3511  2
;  3393     3512  2
;  3394     3513  2      !-----------------------------------------------------------------------
;  3395     3514  2      !
```

```
; 3396     3515  2      ! The first task is to map the Debug Symbol Table (DST) into virtual
; 3397     3516  2      ! address space.  To do this, we must locate the DST in the executable
; 3398     3517  2      ! image file (via the image header),
; 3399     3518  2      ! open the file, call the Create Map Section system service for each symbol
; 3400     3519  2      ! table, and then close the file.  Thereafter the DST can be refer-
; 3401     3520  2      ! enced directly by the other symbol table access routines.
; 3402     3521  2      ! (We'll do the same thing to GST in DBG$GST_INIT, the virtual address
; 3403     3522  2      !  space is released after the GST_RST build.)
; 3404     3523  2      !----------------------------------------------------------------------
; 3405     3524  2      !
; 3406     3525  2      !
; 3407     3526  2      ! Initialize some pointers in OWN or GLOBAL memory to zeroes.
; 3408     3527  2      !
; 3409     3528  2      RST$START_ADDR = 0;
; 3410     3529  2      SAT$START_ADDR = 0;
; 3411     3530  2      DST$BEGIN_ADDR = 0;
; 3412     3531  2      DST$END_ADDR = 0;
; 3413     3532  2
; 3414     3533  2
; 3415     3534  2      ! Start by doing some consistency checks to ensure that the image header
; 3416     3535  2      ! pointer is non-zero and that the debug symbol-table offset in the header
; 3417     3536  2      ! is non-zero.  If either is wrong, signal an error and give up.
; 3418     3537  2      !
; 3419     3538  2      IF .IMG_HEADER EQL 0
; 3420     3539  2      THEN
; 3421     3540  3          BEGIN
; 3422     3541  3          SIGNAL(DBG$_NORSTBLD);
; 3423     3542  3          RETURN;
; 3424     3543  2          END;
; 3425     3544  2
; 3426     3545  2      IF .IMG_HEADER[IHD$W_SYMDBGOFF] EQL 0
; 3427     3546  2      THEN
; 3428     3547  3          BEGIN
; 3429     3548  3          SIGNAL(DBG$_NORSTBLD);
; 3430     3549  3          RETURN;
; 3431     3550  2          END;
; 3432     3551  2
; 3433     3552  2
; 3434     3553  2      ! We have a debug symbol table descriptor in the image header.  If no DST
; 3435     3554  2      ! exists (because all the modules were compiled without /DEBUG or /TRACE),
; 3436     3555  2      ! we indicate this by leaving the DST address zeroed.  Otherwise we set the
; 3437     3556  2      ! DST address to 1 temporarily.
; 3438     3557  2      !
; 3439     3558  2      IMG_DST_BLOCK = .IMG_HEADER + .IMG_HEADER[IHD$W_SYMDBGOFF];
; 3440     3559  2      DSTBLK_COUNT = .IMG_DST_BLOCK[IHS$W_DSTBLKS];
; 3441     3560  2      IF .DSTBLK_COUNT NEQ 0 THEN DST$BEGIN_ADDR = 1;
; 3442     3561  2
; 3443     3562  2
; 3444     3563  2      ! If the DST's Virtual Block Number (VNB) is invalid, we simply pretend that
; 3445     3564  2      ! there is no DST.  (This is actually an error in the image header.)
; 3446     3565  2      !
; 3447     3566  2      IF .IMG_DST_BLOCK[IHS$L_DSTVBN] LEQ 2 THEN DST$BEGIN_ADDR = 0;
; 3448     3567  2
; 3449     3568  2
; 3450     3569  2      ! If this is a Testable Debugger, determine the length of the DST for the
; 3451     3570  2      ! test program, i.e. before the Debugger's own DST records.  Set the size
; 3452     3571  2      ! of the DST in file blocks accordingly so that only the test program's DST
```

```
;  3453     3572  2      ! is mapped into virtual memory below.
;  3454     3573  2      !
;  3455     3574  3      IF (.DBG$GV_CONTROL[DBG$V_CONTROL_TDBG]) AND (.DST$BEGIN_ADDR NEQ 0)
;  3456     3575  2      THEN
;  3457     3576  3          BEGIN
;  3458     3577  3          TESTDSTSIZE = TEST_DST_SIZE(.FILENAME, .IMG_DST_BLOCK);
;  3459     3578  3          DSTBLK_COUNT = (.TESTDSTSIZE + 511)/512;
;  3460     3579  2          END;
;  3461     3580  2
;  3462     3581  2
;  3463     3582  2      ! If we decided above that there is no DST in the executable image, we print
;  3464     3583  2      ! a message saying that there are no local symbols.
;  3465     3584  2      !
;  3466     3585  2      IF .DST$BEGIN_ADDR EQL 0
;  3467     3586  2      THEN
;  3468     3587  2          SIGNAL(DBG$_NOLOCALS)
;  3469     3588  2
;  3470     3589  2
;  3471     3590  2      ! Otherwise we decided that there is a DST, so we create a map section which
;  3472     3591  2      ! maps the DST into our virtual address space.  If this fails, we print the
;  3473     3592  2      ! status message (with informational severity so we can continue) and set
;  3474     3593  2      ! the DST start address to zero; if this succeeds, we set the DST start and
;  3475     3594  2      ! end addresses to the values assigned by $CREMAPSEC.
;  3476     3595  2      !
;  3477     3596  2      ELSE
;  3478     3597  3          BEGIN
;  3479     3598  3          EXE_SECBOUNDS[0] = 200;
;  3480     3599  3          EXE_SECBOUNDS[1] = 1^30;
;  3481     3600  3          EXE_SECNAM_DESC[0] = 3;
;  3482     3601  3          EXE_SECNAM_DESC[1] = UPLIT BYTE(%ASCIC 'DST');
;  3483     3602  3          STATUS = SYS$CRMPSC(EXE_SECBOUNDS, EXE_SECBOUNDS, 0, SEC$M_EXPREG,
;  3484     3603  3                              EXE_SECNAM_DESC, 0, 0,
;  3485     3604  3                              .IMGFILCHAN, .DSTBLK_COUNT,
;  3486     3605  3                              .IMG_DST_BLOCK[IHS$L_DSTVBN], 0, 0);
;  3487     3606  3          IF NOT .STATUS
;  3488     3607  3          THEN
;  3489     3608  4              BEGIN
;  3490     3609  4              DST$BEGIN_ADDR = 0;
;  3491     3610  4              STATUS[STS$V_SEVERITY] = SYS$K_INFO;
;  3492     3611  4              SIGNAL(.STATUS);
;  3493     3612  4              END
;  3494     3613  4
;  3495     3614  3          ELSE
;  3496     3615  4              BEGIN
;  3497     3616  4              DST$BEGIN_ADDR = .EXE_SECBOUNDS[0];
;  3498     3617  4              DST$END_ADDR = .EXE_SECBOUNDS[1];
;  3499     3618  4              IF .DBG$GV_CONTROL[DBG$V_CONTROL_TDBG]
;  3500     3619  4              THEN
;  3501     3620  4                  DST$END_ADDR = .DST$BEGIN_ADDR + .TESTDSTSIZE - 1;
;  3502     3621  4
;  3503     3622  4              DST$NEXT_ADDR = .DST$BEGIN_ADDR;
;  3504     3623  3              END;
;  3505     3624  3
;  3506     3625  2          END;
;  3507     3626  2
;  3508     3627  2
;  3509     3628  2      !------------------------------------------------------------------------
```

```
;  3510          3629  2      !
;  3511          3630  2      ! We have now mapped the DST into our (and the user's) virtual
;  3512          3631  2      ! address space.  Next we want to scan the DST to find all modules in the
;  3513          3632  2      ! program and to build the Run-Time Symbol Table's (the RST's) Module Chain.
;  3514          3633  2      ! We will also build the Program Static Address Table (SAT) which indicates
;  3515          3634  2      ! which static addresses belong to which modules, and we will build the RST
;  3516          3635  2      ! hash table.  Finally, we will determine how much RST space we need for the
;  3517          3636  2      ! several largest modules and allocate enough free space so that their RSTs
;  3518          3637  2      ! can be in memory at the same time.
;  3519          3638  2      !
;  3520          3639  2      !----------------------------------------------------------------------------
;  3521          3640  2      !
;  3522          3641  2      !
;  3523          3642  2      ! First create a Module RST Entry for the "anonymous module" to which we
;  3524          3643  2      ! assign all global symbols (from the GST) which do not belong elsewhere.
;  3525          3644  2      ! Global symbols are attached to this module's symbol chain only as they
;  3526          3645  2      ! are referenced, not now when the RST is initially built.
;  3527          3646  2      !
;  3528          3647  2      RSTPTR = GET_MEM(RST$K_MODENTSIZ);
;  3529          3648  2      RSTPTR[RST$B_KIND] = RST$K_MODULE;
;  3530          3649  2      RSTPTR[RST$V_ANONMOD] = TRUE;
;  3531          3650  2      RSTPTR[RST$B_LANGUAGE] = DBG$K_MACRO;
;  3532          3651  2
;  3533          3652  2
;  3534          3653  2      ! Put the anonymous module at the start of the Module Entry RST chain.
;  3535          3654  2      !
;  3536          3655  2      RST$START_ADDR = .RSTPTR;
;  3537          3656  2      PREV_RST_PTR = .RSTPTR;
;  3538          3657  2
;  3539          3658  2
;  3540          3659  2      ! Initialize the RST Peference List so that RST$REF_LIST points to a list
;  3541          3660  2      ! of zero RST entries.  This list is used by DBG$STA_GETSYMBOL to mark which
;  3542          3661  2      ! RST entries are used by the current Debug command and thus cannot be re-
;  3543          3662  2      ! moved from the RST until the command ends.
;  3544          3663  2      !
;  3545          3664  2      RST$REF_LIST = GET_MEM(22);
;  3546          3665  2      RST$REF_LIST[0] = 20;
;  3547          3666  2      RST$REF_LIST[1] = 0;
;  3548          3667  2
;  3549          3668  2
;  3550          3669  2      ! Create the list head for the Least Recently Used Module (LRUM) table.
;  3551          3670  2      ! This table is used by DBG$GET_MEMORY to determine what module to remove
;  3552          3671  2      ! from the RST when more free memory space is needed.
;  3553          3672  2      !
;  3554          3673  2      LRUM$LISTHEAD = GET_MEM(LRUM$ENTSIZE);
;  3555          3674  2      LRUM$LISTHEAD[LRUM$L_FLINK] = .LRUM$LISTHEAD;
;  3556          3675  2      LRUM$LISTHEAD[LRUM$L_BLINK] = .LRUM$LISTHEAD;
;  3557          3676  2      LRUM$LISTHEAD[LRUM$L_RSTPTR] = 0;
;  3558          3677  2
;  3559          3678  2
;  3560          3679  2      ! Set up all Source File Control Blocks and associated RMS blocks that we
;  3561          3680  2      ! need to do source line display.
;  3562          3681  2      !
;  3563          3682  2      DBG$SRC_INIT();
;  3564          3683  2
;  3565          3684  2
;  3566          3685  2      ! Initialize the Screen Debugging feature.
```

```
;  3567          3686  2     !
;  3568          3687  2     DBG$SCR_INITIALIZE();
;  3569          3688  2
;  3570          3689  2
;  3571          3690  2     ! Zero out all entries in the Module Size Table.  This table will give the
;  3572          3691  2     ! RST space requirements of the several largest modules in the DST.  Also
;  3573          3692  2     ! zero out several other counters and flags.
;  3574          3693  2     !
;  3575          3694  2     INCR I FROM 0 TO MIN_MODS_IN_RST - 1 DO
;  3576          3695  2         MOD_SIZE_TBL[.I] = 0;
;  3577          3696  2
;  3578          3697  2     IN_MODULE_FLAG = FALSE;
;  3579          3698  2     ERROR_COUNT = 0;
;  3580          3699  2     REC_NESTING = 0;
;  3581          3700  2     SAT$START_ADDR = 0;
;  3582          3701  2
;  3583          3702  2
;  3584          3703  2     ! If this executable image was linked by a VMS V4.0 or later linker, then
;  3585          3704  2     ! we use the Debug Module Table to initialize the RST.  This is a much
;  3586          3705  2     ! faster way of doing the initialization than reading through the whole
;  3587          3706  2     ! Debug Symbol Table.  If this succeeds, we set up DSTPTR so that the loop
;  3588          3707  2     ! below to scan the whole DST is skipped.
;  3589          3708  2     !
;  3590          3709  2     DSTPTR = .DST$BEGIN_ADDR;
;  3591          3710  2     IF .(IMG_HEADER[IHD$L_LNKFLAGS])<V_(5)> AND
;  3592          3711  3         (.DSTPTR[DST$L_MODBEG_LANGUAGE] NEQ DST$K_BASIC)
;  3593          3712  2     THEN
;  3594          3713  3         BEGIN
;  3595          3714  3         STATUS = READ_DEBUG_MODULE_TABLE(.IMGFILCHAN,
;  3596          3715  3                                          .IMG_DST_BLOCK, MOD_SIZE_TBL[0]);
;  3597          3716  3         IF .STATUS THEN DSTPTR = .DST$END_ADDR;
;  3598          3717  2         END;
;  3599          3718  2
;  3600          3719  2
;  3601          3720  2     ! Scan the Debug Symbol Table to find all modules in it.  For each module,
;  3602          3721  2     ! determine how much RST space that module's symbol table will require when
;  3603          3722  2     ! when brought into the RST.  Also build its Static Address Table entries.
;  3604          3723  2     ! Then create its Module RST Entry and put it on the RST Module Chain.
;  3605          3724  2     ! Note that all this is done only if we did not initialize the RST from the
;  3606          3725  2     ! the Debug Module Table, i.e. only if this is an old executable image.
;  3607          3726  2     !
;  3608          3727  2     WHILE .DSTPTR LSSU .DST$END_ADDR DO
;  3609          3728  3         BEGIN
;  3610          3729  3
;  3611          3730  3
;  3612          3731  3         ! Pick up the DST record's type and use it as a CASE index so each
;  3613          3732  3         ! significant kind of DST record can be processed individually.
;  3614          3733  3         !
;  3615          3734  3         IF .DSTPTR[DST$B_LENGTH] EQL 0
;  3616          3735  3         THEN
;  3617          3736  3             TYPE = 257
;  3618          3737  3         ELSE
;  3619          3738  3             TYPE = .DSTPTR[DST$B_TYPE];
;  3620          3739  3         IF (NOT .IN_MODULE_FLAG) AND (.TYPE NEQ DST$K_MODBEG) THEN EXITLOOP;
;  3621          3740  3         CASE .TYPE FROM 0 TO 257 OF
;  3622          3741  3             SET
;  3623          3742  3
```

```
;  3624     3743  3
;  3625     3744  3          ! Handle the error case where we have a zero length DST record
;  3626     3745  3          ! within a module. A zero length DST record outside a module
;  3627     3746  3          ! is OK.
;  3628     3747  3          !
;  3629     3748  3          [257]:
;  3630     3749  4              BEGIN
;  3631     3750  4              SIGNAL(DBG$_ZERLENDST);
;  3632     3751  3              END;
;  3633     3752  3
;  3634     3753  3
;  3635     3754  3          ! Handle the error case where we have a DST record not nested within
;  3636     3755  3          ! a Module-Begin/Module-End pair.  Give an error message, but quit
;  3637     3756  3          ! printing the message if we have done so too many times already.
;  3638     3757  3          !
;  3639     3758  3          [256]:
;  3640     3759  4              BEGIN
;  3641     3760  4              IF .ERROR_COUNT LSS 50 THEN SIGNAL(DBG$_MISMODBEG);
;  3642     3761  4              IF .ERROR_COUNT EQL 50 THEN SIGNAL(DBG$_TOOMANERR);
;  3643     3762  4              ERROR_COUNT = .ERROR_COUNT + 1;
;  3644     3763  3              END;
;  3645     3764  3
;  3646     3765  3
;  3647     3766  3          ! Handle the Module Begin record.  Here we make sure we are not al-
;  3648     3767  3          ! ready in a module (i.e., we check for nesting errors), and then
;  3649     3768  3          ! we set up all pointers and counters we will need to collect infor-
;  3650     3769  3          ! mation about this module and later create a Module RST Entry.
;  3651     3770  3          !
;  3652     3771  3          [DST$K_MODBEG]:
;  3653     3772  4              BEGIN
;  3654     3773  4              IF .IN_MODULE_FLAG THEN SIGNAL(DBG$_MISMODEND, 1, .MODNAMEPTR);
;  3655     3774  4              IN_MODULE_FLAG = TRUE;
;  3656     3775  4              MOD_DSTPTR = .DSTPTR;
;  3657     3776  4              MODNAMEPTR = MOD_DSTPTR[DST$B_MODBEG_NAME];
;  3658     3777  4              NUM_RST_ENTS = 0;
;  3659     3778  4              NUM_RST_LONGS = 0;
;  3660     3779  4              NUM_SAT_ENTS = 0;
;  3661     3780  4              TMP_SAT_CHAIN = 0;
;  3662     3781  4              ROUT_STACK_INDEX = 0;
;  3663     3782  4              LANGUAGE = .MOD_DSTPTR[DST$L_MODBEG_LANGUAGE];
;  3664     3783  3              END;
;  3665     3784  3
;  3666     3785  3
;  3667     3786  3          ! Handle the Module End record.  Build the Module RST Entry for this
;  3668     3787  3          ! module and link it into the RST Module Chain.  Then see if it is
;  3669     3788  3          ! among the several largest modules in RST space requirements--if so
;  3670     3789  3          ! we remember how much RST space it needs.  Also copy the Static
;  3671     3790  3          ! Address Table entries on this module's temporary SAT chain over to
;  3672     3791  3          ! the permanent Program Static Address Table chain.
;  3673     3792  3          !
;  3674     3793  3          [DST$K_MODEND]:
;  3675     3794  4              BEGIN
;  3676     3795  4              IN_MODULE_FLAG = FALSE;
;  3677     3796  4
;  3678     3797  4
;  3679     3798  4              ! Determine how much RST space the module needs for its symbol
;  3680     3799  4              ! table.  Note that this includes both the space needed for RST
```

```
;  3681          3800  4                ! and SAT entries plus one longword for memory allocation con-
;  3682          3801  4                ! trol for each such entry.
;  3683          3802  4                !
;  3684          3803  4                RSTSIZE = .NUM_RST_ENTS + .NUM_RST_LONGS
;  3685          3804  4                        + .NUM_SAT_ENTS*(1 + SAT$K_ENTSIZE);
;  3686          3805  4
;  3687          3806  4
;  3688          3807  4                ! Build the Module RST Entry for the current module.
;  3689          3808  4                !
;  3690          3809  4                RSTPTR = GET_MEM(RST$K_MODENTSIZ);
;  3691          3810  4                RSTPTR[RST$L_DSTPTR] = .MOD_DSTPTR;
;  3692          3811  4                RSTPTR[RST$B_KIND] = RST$K_MODULE;
;  3693          3812  4                RSTPTR[RST$L_MODRSTSIZ] = 4*.RSTSIZE;
;  3694          3813  4                RSTPTR[RST$B_LANGUAGE] = .LANGUAGE;
;  3695          3814  4
;  3696          3815  4
;  3697          3816  4                ! Link the Module RST Entry into the RST Module Chain.
;  3698          3817  4                !
;  3699          3818  4                PREV_RST_PTR[RST$L_NXTMODPTR] = .RSTPTR;
;  3700          3819  4                PREV_RST_PTR = .RSTPTR;
;  3701          3820  4
;  3702          3821  4
;  3703          3822  4                ! If the amount of RST space the module needs makes it one of
;  3704          3823  4                ! the several largest modules, enter it in MOD_SIZE_TBL.  Note
;  3705          3824  4                ! that the table remains sorted on module size, largest first.
;  3706          3825  4                !
;  3707          3826  4                INCR I FROM 0 TO MIN_MODS_IN_RST - 1 DO
;  3708          3827  5                    BEGIN
;  3709          3828  5                    IF .RSTSIZE GTR .MOD_SIZE_TBL[.I]
;  3710          3829  5                    THEN
;  3711          3830  6                        BEGIN
;  3712          3831  6                        DECR J FROM MIN_MODS_IN_RST - 2 TO .I DO
;  3713          3832  6                            MOD_SIZE_TBL[.J+1] = .MOD_SIZE_TBL[.J];
;  3714          3833  6
;  3715          3834  6                        MOD_SIZE_TBL[.I] = .RSTSIZE;
;  3716          3835  6                        EXITLOOP;
;  3717          3836  5                        END;
;  3718          3837  5
;  3719          3838  4                    END;
;  3720          3839  4
;  3721          3840  4
;  3722          3841  4                ! Copy the Static Address Table entries we have built and merged
;  3723          3842  4                ! on the TMP_SAT_CHAIN chain (which covers the current module
;  3724          3843  4                ! only) over to the permanent Program Static Address Table.
;  3725          3844  4                !
;  3726          3845  4                PREV_SAT_PTR = SAT$START_ADDR;
;  3727          3846  4                PROG_SAT_PTR = .SAT$START_ADDR;
;  3728          3847  4                TMP_SAT_PTR = .TMP_SAT_CHAIN;
;  3729          3848  4                WHILE .TMP_SAT_PTR NEQ 0 DO
;  3730          3849  5                    BEGIN
;  3731          3850  5
;  3732          3851  5
;  3733          3852  5                    ! Find the proper place in the permanent chain to insert the
;  3734          3853  5                    ! new entry so the chain remains sorted on start address.
;  3735          3854  5                    !
;  3736          3855  5                    WHILE .PROG_SAT_PTR NEQ 0 DO
;  3737          3856  6                        BEGIN
```

```
;  3738     3857  6                    IF .PROG_SAT_PTR[SAT$L_START] GTRU .TMP_SAT_PTR[SAT$L_START]
;  3739     3858  6                    THEN
;  3740     3859  6                        EXITLOOP;
;  3741     3860  6
;  3742     3861  6                    PREV_SAT_PTR = .PROG_SAT_PTR;
;  3743     3862  6                    PROG_SAT_PTR = .PROG_SAT_PTR[SAT$L_FLINK];
;  3744     3863  5                    END;
;  3745     3864  5
;  3746     3865  5
;  3747     3866  5                ! Fill the module RST pointer into the new SAT entry and
;  3748     3867  5                ! link it into the permanent Program SAT.  Then step to
;  3749     3868  5                ! the next TMP_SAT_CHAIN entry.
;  3750     3869  5                !
;  3751     3870  5                NEXTSAT = .TMP_SAT_PTR[SAT$L_FLINK];
;  3752     3871  5                TMP_SAT_PTR[SAT$L_RSTPTR] = .RSTPTR;
;  3753     3872  5                TMP_SAT_PTR[SAT$L_FLINK] = .PROG_SAT_PTR;
;  3754     3873  5                PREV_SAT_PTR[SAT$L_FLINK] = .TMP_SAT_PTR;
;  3755     3874  5                PREV_SAT_PTR = .TMP_SAT_PTR;
;  3756     3875  5                TMP_SAT_PTR = .NEXTSAT;
;  3757     3876  4                END;
;  3758     3877  4
;  3759     3878  3            END;                    ! End of Module-End-record code
;  3760     3879  3
;  3761     3880  3
;  3762     3881  3        ! Handle the PSECT record.  For each PSECT record we record its
;  3763     3882  3        ! address range in the temporary Static Address Table we are build-
;  3764     3883  3        ! ing for this module.  (That table will be copied to the Program
;  3765     3884  3        ! Static Address Table when the Module-End record is found.)  We
;  3766     3885  3        ! also tabulate that the PSECT will require a Label RST Entry.
;  3767     3886  3        !
;  3768     3887  3        [DST$K_PSECT]:
;  3769     3888  4            BEGIN
;  3770     3889  4            STARTADDR = .DSTPTR[DST$L_PSECT_VALUE];
;  3771     3890  4            PPTR = DSTPTR[DST$A_PSECT_TRLR_BASE]
;  3772     3891  4                    + .DSTPTR[DST$B_PSECT_TRLR_OFFS];
;  3773     3892  4            LENGTH = .PPTR[DST$L_PSECT_SIZE];
;  3774     3893  4            STATIC_ADDRESS(.STARTADDR, .LENGTH, 0, TMP_SAT_CHAIN, TRUE);
;  3775     3894  4            NUM_SAT_ENTS = .NUM_SAT_ENTS + 1;
;  3776     3895  4            NUM_RST_ENTS = .NUM_RST_ENTS + 1;
;  3777     3896  4            NUM_RST_LONGS = .NUM_RST_LONGS + RST$K_LBLENTSIZ;
;  3778     3897  3            END;
;  3779     3898  3
;  3780     3899  3
;  3781     3900  3        ! Handle the Routine-Begin record.  Tabulate how much RST space it
;  3782     3901  3        ! requires.  Also remember the routine start address in the routine
;  3783     3902  3        ! start address stack--we will need it when we find the Routine-End
;  3784     3903  3        ! record.
;  3785     3904  3        !
;  3786     3905  3        [DST$K_RTNBEG]:
;  3787     3906  4            BEGIN
;  3788     3907  4            NUM_RST_ENTS = .NUM_RST_ENTS + 1;
;  3789     3908  4            NUM_RST_LONGS = .NUM_RST_LONGS + RST$K_ROUTENTSIZ;
;  3790     3909  4            NUM_SAT_ENTS = .NUM_SAT_ENTS + 1;
;  3791     3910  4            IF .ROUT_STACK_INDEX LSS MAX_ROUT_STACK
;  3792     3911  4            THEN
;  3793     3912  4                ROUT_STACK[.ROUT_STACK_INDEX] = .DSTPTR[DST$L_RTNBEG_ADDRESS]
;  3794     3913  4
```

```
;  3795     3914  4              ELSE IF .LANGUAGE NEQ DBG$K_MACRO
;  3796     3915  4              THEN
;  3797     3916  4                  SIGNAL(DBG$_DSTNESDEP, 1, .MODNAMEPTR);
;  3798     3917  4
;  3799     3918  4              ROUT_STACK_INDEX = .ROUT_STACK_INDEX + 1;
;  3800     3919  3              END;
;  3801     3920  3
;  3802     3921  3
;  3803     3922  3            ! Handle the Routine-End record.  Determine what address range the
;  3804     3923  3            ! routine body covers and enter that range in the temporary Static
;  3805     3924  3            ! Address Table we are building for the current module.  (That info
;  3806     3925  3            ! will be copied to the Program Static Address Table when we find
;  3807     3926  3            ! the Module-End record.)
;  3808     3927  3            !
;  3809     3928  3            [DST$K_RTNEND]:
;  3810     3929  4                BEGIN
;  3811     3930  4                ROUT_STACK_INDEX = .ROUT_STACK_INDEX - 1;
;  3812     3931  4                IF .ROUT_STACK_INDEX LSS 0
;  3813     3932  4                THEN
;  3814     3933  4                    SIGNAL(DBG$_INCDSTNES, 1, .MODNAMEPTR)
;  3815     3934  4
;  3816     3935  4                ELSE IF .ROUT_STACK_INDEX LSS MAX_ROUT_STACK
;  3817     3936  4                THEN
;  3818     3937  5                    BEGIN
;  3819     3938  5                    STARTADDR = .ROUT_STACK[.ROUT_STACK_INDEX];
;  3820     3939  5                    LENGTH = .DSTPTR[DST$L_RTNEND_SIZE];
;  3821     3940  5                    STATIC_ADDRESS(.STARTADDR, .LENGTH, 0, TMP_SAT_CHAIN, TRUE);
;  3822     3941  4                    END;
;  3823     3942  4
;  3824     3943  3                END;
;  3825     3944  3
;  3826     3945  3
;  3827     3946  3            ! Handle the Lexical Block-Begin record.  Tabulate how much RST
;  3828     3947  3            ! space it will require.
;  3829     3948  3            !
;  3830     3949  3            [DST$K_BLKBEG]:
;  3831     3950  4                BEGIN
;  3832     3951  4                NUM_RST_ENTS = .NUM_RST_ENTS + 1;
;  3833     3952  4                NUM_RST_LONGS = .NUM_RST_LONGS + RST$K_LEXENTSIZ;
;  3834     3953  4                NUM_SAT_ENTS = .NUM_SAT_ENTS + 1;
;  3835     3954  3                END;
;  3836     3955  3
;  3837     3956  3
;  3838     3957  3            ! Handle the Label record.  Again just tabulate how much RST space
;  3839     3958  3            ! is required for this record.
;  3840     3959  3            !
;  3841     3960  3            [DST$K_LABEL]:
;  3842     3961  4                BEGIN
;  3843     3962  4                NUM_RST_ENTS = .NUM_RST_ENTS + 1;
;  3844     3963  4                NUM_RST_LONGS = .NUM_RST_LONGS + RST$K_LBLENTSIZ;
;  3845     3964  4                NUM_SAT_ENTS = .NUM_SAT_ENTS + 1;
;  3846     3965  3                END;
;  3847     3966  3
;  3848     3967  3
;  3849     3968  3            ! Handle the Label-or-Literal record.  If the record is for a Label,
;  3850     3969  3            ! tabulate the space needed for a Label RST Entry and a SAT entry.
;  3851     3970  3            ! If it is for a literal, tabulate the space needed for a Data Item
```

```
;  3852     3971  3              ! RST Entry.
;  3853     3972  3              !
;  3854     3973  3              [DST$K_LBLORLIT]:
;  3855     3974  4                  BEGIN
;  3856     3975  4                  NUM_RST_ENTS = .NUM_RST_ENTS + 1;
;  3857     3976  4                  IF .DSTPTR[DST$B_VFLAGS] EQL 1
;  3858     3977  4                  THEN
;  3859     3978  5                      BEGIN
;  3860     3979  5                      NUM_RST_LONGS = .NUM_RST_LONGS + RST$K_LBLENTSIZ;
;  3861     3980  5                      NUM_SAT_ENTS = .NUM_SAT_ENTS + 1;
;  3862     3981  5                      END
;  3863     3982  5
;  3864     3983  4                  ELSE
;  3865     3984  4                      NUM_RST_LONGS = .NUM_RST_LONGS + RST$K_DATENTSIZ;
;  3866     3985  4
;  3867     3986  3                  END;
;  3868     3987  3
;  3869     3988  3
;  3870     3989  3              ! Handle the Entry-Point record.  Tabulate the RST space it needs.
;  3871     3990  3              !
;  3872     3991  3              [DST$K_ENTRY]:
;  3873     3992  4                  BEGIN
;  3874     3993  4                  NUM_RST_ENTS = .NUM_RST_ENTS + 1;
;  3875     3994  4                  NUM_RST_LONGS = .NUM_RST_LONGS + RST$K_EPTENTSIZ;
;  3876     3995  4                  NUM_SAT_ENTS = .NUM_SAT_ENTS + 1;
;  3877     3996  3                  END;
;  3878     3997  3
;  3879     3998  3
;  3880     3999  3              ! Handle the Record-Begin record.  Tabulate the space needed for the
;  3881     4000  3              ! Type RST Entry for the record and a possible Data Item RST Entry.
;  3882     4001  3              ! Also increment the current level of record nesting.
;  3883     4002  3              !
;  3884     4003  3              [DST$K_RECBEG]:
;  3885     4004  4                  BEGIN
;  3886     4005  4                  NUM_RST_ENTS = .NUM_RST_ENTS + 2;
;  3887     4006  4                  NUM_RST_LONGS = .NUM_RST_LONGS +
;  3888     4007  4                                  RST$K_TYPENTSIZ + RST$K_DATENTSIZ + 1;
;  3889     4008  4                  IF .REC_NESTING GTR 0 THEN NUM_RST_LONGS = .NUM_RST_LONGS + 1;
;  3890     4009  4                  REC_NESTING = .REC_NESTING + 1;
;  3891     4010  4                  STATIC_CHECK(.DSTPTR, STARTADDR, LENGTH, STATIC_FLAG);
;  3892     4011  4                  IF .STATIC_FLAG
;  3893     4012  4                  THEN
;  3894     4013  5                      BEGIN
;  3895     4014  5                      STATIC_ADDRESS(.STARTADDR, .LENGTH, 0, TMP_SAT_CHAIN, TRUE);
;  3896     4015  5                      NUM_SAT_ENTS = .NUM_SAT_ENTS + 1;
;  3897     4016  4                      END;
;  3898     4017  3                  END;
;  3899     4018  3
;  3900     4019  3
;  3901     4020  3              ! Handle the Record End record.  Just decrement the record nesting
;  3902     4021  3              ! depth by one.
;  3903     4022  3              !
;  3904     4023  3              [DST$K_RECEND]:
;  3905     4024  4                  BEGIN
;  3906     4025  4                  IF .REC_NESTING LEQ 0 THEN SIGNAL(DBG$_INCDSTNES,1,.MODNAMEPTR);
;  3907     4026  4                  REC_NESTING = .REC_NESTING - 1;
;  3908     4027  3                  END;
```

```
;  3909          4028  3
;  3910          4029  3
;  3911          4030  3              ! Handle the Variant-Set Begin record.  Tabulate the space needed
;  3912          4031  3              ! for one Variant RST Entry plus a component list pointer to it.
;  3913          4032  3              !
;  3914          4033  3              [DST$K_VARBEG]:
;  3915          4034  3                  NUM_RST_LONGS = .NUM_RST_LONGS + RST$K_VARENTSIZ + 1;
;  3916          4035  3
;  3917          4036  3
;  3918          4037  3              ! Handle the Variant Value record.  Note that we need 2 longwords
;  3919          4038  3              ! per variant entry plus one more for a list pointer.
;  3920          4039  3              !
;  3921          4040  3              [DST$K_VARVAL]:
;  3922          4041  3                  NUM_RST_LONGS = .NUM_RST_LONGS + 3;
;  3923          4042  3
;  3924          4043  3
;  3925          4044  3              ! Handle the Enumeration Type Begin record.  Tabulate the space
;  3926          4045  3              ! needed for a Data Type RST Entry.
;  3927          4046  3              !
;  3928          4047  3              [DST$K_ENUMBEG]:
;  3929          4048  4                  BEGIN
;  3930          4049  4                  NUM_RST_ENTS = .NUM_RST_ENTS + 1;
;  3931          4050  4                  NUM_RST_LONGS = .NUM_RST_LONGS + RST$K_TYPENTSIZ;
;  3932          4051  3                  END;
;  3933          4052  3
;  3934          4053  3
;  3935          4054  3              ! Handle the Enumeration Type Element record.  Tabulate the space
;  3936          4055  3              ! needed for one Data RST Entry plus a longword pointer to it.
;  3937          4056  3              !
;  3938          4057  3              [DST$K_ENUMELT]:
;  3939          4058  4                  BEGIN
;  3940          4059  4                  NUM_RST_ENTS = .NUM_RST_ENTS + 1;
;  3941          4060  4                  NUM_RST_LONGS = .NUM_RST_LONGS + RST$K_DATENTSIZ + 1;
;  3942          4061  3                  END;
;  3943          4062  3
;  3944          4063  3
;  3945          4064  3              ! Handle all Data Symbol records.  This includes the Bliss special
;  3946          4065  3              ! cases, the Boolean record, and the Separate Type Spec record.  For
;  3947          4066  3              ! each, tabulate the amount of RST space needed for the data symbol
;  3948          4067  3              ! and make a Static Address Table entry if this is a static data
;  3949          4068  3              ! item.
;  3950          4069  3              !
;  3951          4070  3              [DSC$K_DTYPE_LOWEST TO DSC$K_DTYPE_HIGHEST,
;  3952          4071  3               DST$K_BLI,
;  3953          4072  3               DST$K_BOOL,
;  3954          4073  3               DST$K_SEPTYP]:
;  3955          4074  4                  BEGIN
;  3956          4075  4                  NUM_RST_ENTS = .NUM_RST_ENTS + 1;
;  3957          4076  4                  NUM_RST_LONGS = .NUM_RST_LONGS + RST$K_DATENTSIZ;
;  3958          4077  4                  IF .REC_NESTING GTR 0 THEN NUM_RST_LONGS = .NUM_RST_LONGS + 1;
;  3959          4078  4                  STATIC_CHECK(.DSTPTR, STARTADDR, LENGTH, STATIC_FLAG);
;  3960          4079  4                  IF .STATIC_FLAG
;  3961          4080  4                  THEN
;  3962          4081  5                      BEGIN
;  3963          4082  5                      STATIC_ADDRESS(.STARTADDR, .LENGTH, 0, TMP_SAT_CHAIN, TRUE);
;  3964          4083  5                      NUM_SAT_ENTS = .NUM_SAT_ENTS + 1;
;  3965          4084  4                      END;
```

```
;  3966          4085  4
;  3967          4086  3                END;
;  3968          4087  3
;  3969          4088  3
;  3970          4089  3            ! Handle the BLISS fiel. record.  Here we allow for one Data Item
;  3971          4090  3            ! RST Entry.
;  3972          4091  3            !
;  3973          4092  3            [DST$K_BLIFLD]:
;  3974          4093  4                BEGIN
;  3975          4094  4                NUM_RST_ENTS = .NUM_RST_ENTS + 1;
;  3976          4095  4                NUM_RST_LONGS = .NUM_RST_LONGS + RST$K_DATENTSIZ;
;  3977          4096  4                END;
;  3978          4097  3
;  3979          4098  3
;  3980          4099  3            ! Handle the COBOL special cases.  Tabulate the space needed for one
;  3981          4100  3            ! Data Item RST Entry.
;  3982          4101  3            !
;  3983          4102  3            [DST$K_COB_HACK]:
;  3984          4103  4                BEGIN
;  3985          4104  4                NUM_RST_ENTS = .NUM_RST_ENTS + 1;
;  3986          4105  4                NUM_RST_LONGS = .NUM_RST_LONGS + RST$K_DATENTSIZ;
;  3987          4106  3                END;
;  3988          4107  3
;  3989          4108  3
;  3990          4109  3            ! For all other records, we do nothing.
;  3991          4110  3            !
;  3992          4111  3            [INRANGE]:
;  3993          4112  3                0;
;  3994          4113  3
;  3995          4114  3            TES;
;  3996          4115  3
;  3997          4116  3
;  3998          4117  3            ! Advance the DST pointer to point to the next record.  Then loop.
;  3999          4118  3            !
;  4000          4119  3            DSTPTR = DSTPTR[DST$A_NEXT] + .DSTPTR[DST$B_LENGTH];
;  4001          4120  3
;  4002          4121  2            END;                                ! End of WHILE loop over whole DST
;  4003          4122  2
;  4004          4123  2
;  4005          4124  2        ! If we come out of the loop but we are still in a module then
;  4006          4125  2        ! then the module end record was missing. Signal an error.
;  4007          4126  2        !
;  4008          4127  2        IF .IN_MODULE_FLAG
;  4009          4128  2        THEN
;  4010          4129  2            SIGNAL(DBG$_MISMODEND, 1, .MODNAMEPTR);
;  4011          4130  2
;  4012          4131  2
;  4013          4132  2        ! Now initialize the RST Hash Table and enter all module names in it.
;  4014          4133  2        ! Note that we omit the anonymous module.
;  4015          4134  2        !
;  4016          4135  2        INCR I FROM 0 TO RST$K_HASHTBLSIZE - 1 DO
;  4017          4136  3            BEGIN
;  4018          4137  3            RST$HASH_TABLE[2*.I] = RST$HASH_TABLE[2*.I];
;  4019          4138  3            RST$HASH_TABLE[2*.I + 1] = RST$HASH_TABLE[2*.I];
;  4020          4139  2            END;
;  4021          4140  2
;  4022          4141  2        RSTPTR = .RST$START_ADDR;
```

```
;  4023          4142  2     RSTPTR = .RSTPTR[RST$L_NXTMODPTR];
;  4024          4143  2     WHILE .RSTPTR NEQ 0 DO
;  4025          4144  3         BEGIN
;  4026          4145  3         DBG$HASH_INSERT(.RSTPTR);
;  4027          4146  3         RSTPTR = .RSTPTR[RST$L_NXTMODPTR];
;  4028          4147  2         END;
;  4029          4148  2
;  4030          4149  2
;  4031          4150  2     ! Also create a global symbol SHARE$xxx in the RST for each shared image
;  4032          4151  2     ! xxx in the current execution.  This allows the user to see the start
;  4033          4152  2     ! address of each shared image (actually every image whether shared or
;  4034          4153  2     ! not) in the execution.
;  4035          4154  2     !
;  4036          4155  2     ! Note: Shareable image format in VMS 4.0 (3B) is different from
;  4037          4156  2     ! VMS 3.0.  So we call different routines depending on the system.
;  4038          4157  2     !
;  4039          4158  2     IF .DBG$GV_CONTROL[DBG$V_CONTROL_VERSION_4]
;  4040          4159  2     THEN
;  4041          4160  2         SHARED_IMAGE_NAMES(.IMGFILCHAN)
;  4042          4161  2     ELSE
;  4043          4162  2         SHARED_IMAGE_NAMES_V3(.IMGFILCHAN);
;  4044          4163  2
;  4045          4164  2
;  4046          4165  2     ! Build RST entries for all Global Symbols in GST.
;  4047          4166  2     !
;  4048          4167  2     IF NOT DBG$GST_INIT(.IMGFILCHAN, 0, .RST$START_ADDR)
;  4049          4168  2     THEN
;  4050          4169  2         SIGNAL(DBG$_NOGLOBALS);
;  4051          4170  2
;  4052          4171  2
;  4053          4172  2     ! Determine how much free memory space we need to accommodate the RST's of
;  4054          4173  2     ! the largest several modules plus enough working space.  If we don't have
;  4055          4174  2     ! that much, we expand the free memory pool (via DBG$EXPAND_MEMORY) so that
;  4056          4175  2     ! we will have enough.  Note that if we cannot expand the pool that much
;  4057          4176  2     ! (i.e, if DBG$EXPAND_MEMORY fails because the system cannot come up with
;  4058          4177  2     ! the extra virtual address space), we scale down the request by one module
;  4059          4178  2     ! at a time and try again--hence the WHILE loop here.
;  4060          4179  2     !
;  4061          4180  2     LONGS_NEEDED = 8192;                        ! 32K bytes of working space
;  4062          4181  2     INCR I FROM 0 TO MIN_MODS_IN_RST - 1 DO
;  4063          4182  2         LONGS_NEEDED = .LONGS_NEEDED + .MOD_SIZE_TBL[.I];
;  4064          4183  2
;  4065          4184  2     LONGS_NEEDED = .LONGS_NEEDED - DBG$FREE_MEM_LEFT();
;  4066          4185  2     MST_INDEX = MIN_MODS_IN_RST;
;  4067          4186  2     WHILE .LONGS_NEEDED GTR 100 DO
;  4068          4187  3         BEGIN
;  4069          4188  3         STATUS = DBG$EXPAND_MEMORY(.LONGS_NEEDED);
;  4070          4189  3         IF .STATUS OR (.MST_INDEX EQL 0) THEN EXITLOOP;
;  4071          4190  3         MST_INDEX = .MST_INDEX - 1;
;  4072          4191  3         LONGS_NEEDED = .LONGS_NEEDED - .MOD_SIZE_TBL[.MST_INDEX];
;  4073          4192  2         END;
;  4074          4193  2
;  4075          4194  2
;  4076          4195  2     ! The DST and RST are now initialized, the Program Static Address Table is
;  4077          4196  2     ! built, the RST Hash Table is built, and the necessary free memory has
;  4078          4197  2     ! been acquired (if it was possible).  We now set up the Scope List with
;  4079          4198  2     ! the default scopes.
```

```
;  4080          4199  2     !
;  4081          4200  2     DBG$RST_SETSCOPE(UPLIT(0), RSTPTR);
;  4082          4201  2
;  4083          4202  2
;  4084          4203  2     ! Now search the Static Address Table for the module with the transfer
;  4085          4204  2     ! address so we can SET that module.
;  4086          4205  2     !
;  4087          4206  2     RSTPTR = 0;
;  4088          4207  2     SATPTR = .SAT$START_ADDR;
;  4089          4208  2     WHILE .SATPTR NEQ 0 DO
;  4090          4209  3         BEGIN
;  4091          4210  3         IF .TRANSFER_ADDR GEQ .SATPTR[SAT$L_START] AND
;  4092          4211  3            .TRANSFER_ADDR LEQ .SATPTR[SAT$L_END]
;  4093          4212  3         THEN
;  4094          4213  4             BEGIN
;  4095          4214  4             RSTPTR = .SATPTR[SAT$L_RSTPTR];
;  4096          4215  4             EXITLOOP;
;  4097          4216  3             END;
;  4098          4217  3
;  4099          4218  3         SATPTR = .SATPTR[SAT$L_FLINK];
;  4100          4219  2         END;
;  4101          4220  2
;  4102          4221  2
;  4103          4222  2     ! If we did not find the module with the transfer address, we SET the first
;  4104          4223  2     ! module in the RST (and DST) instead.
;  4105          4224  2     !
;  4106          4225  2     IF .RSTPTR EQL 0
;  4107          4226  2     THEN
;  4108          4227  3         BEGIN
;  4109          4228  3         RSTPTR = .RST$START_ADDR;
;  4110          4229  3         RSTPTR = .RSTPTR[RST$L_NXTMODPTR];
;  4111          4230  2         END;
;  4112          4231  2
;  4113          4232  2
;  4114          4233  2     ! If we are in SUPERDEBUG, look up symbol BEGINHERE in the GST and set its
;  4115          4234  2     ! address in the current PC.  If the lookup fails, we die right here.  Also
;  4116          4235  2     ! set the language to BLISS and do not SET any initial module.
;  4117          4236  2     !
;  4118          4237  2     IF .DBG$GV_CONTROL[DBG$V_CONTROL_SDBG]
;  4119          4238  2     THEN
;  4120          4239  3         BEGIN
;  4121          4240  3         BIND BEGINHERENAME = UPLIT BYTE(%ASCIC 'BEGINHERE');
;  4122          4241  3         RSTPTR = DBG$STA_LOOKUP_GBL(BEGINHERENAME);
;  4123          4242  3         IF .RSTPTR EQL 0 THEN SIGNAL(DBG$_NOSYMBOL, 1, BEGINHERENAME);
;  4124          4243  3         DSTPTR = .RSTPTR[RST$L_DSTPTR];
;  4125          4244  3         DBG$RUNFRAME[DBG$L_USER_PC] = .DSTPTR[DST$L_VALUE] + 2;
;  4126          4245  3         DBG$SET_LANG(0, DBG$K_BLISS);
;  4127          4246  3         RSTPTR = 0;
;  4128          4247  2         END;
;  4129          4248  2
;  4130          4249  2
;  4131          4250  2     ! Now actually SET the selected module and build its RST.  Also print the
;  4132          4251  2     ! start-up message and set the language code accordingly.  Note that the
;  4133          4252  2     ! module must be SET before we can use its language code--the language code
;  4134          4253  2     ! is not set if READ_DEBUG_MODULE_TABLE did the RST initialization.
;  4135          4254  2     !
;  4136          4255  2     IF .RSTPTR NEQ 0
```

```
;  4137          4256  2     THEN
;  4138          4257  3         BEGIN
;  4139          4258  3         RSTPTR[RST$V_MODSET] = TRUE;
;  4140          4259  3         DBG$RST_BUILD(.RSTPTR, TRUE);
;  4141          4260  3         DBG$SET_LANG(0, .RSTPTR[RST$B_LANGUAGE]);
;  4142          4261  3         DSTPTR = .RSTPTR[RST$L_DSTPTR];
;  4143          4262  3         SIGNAL(DBG$_INITIAL, 2,
;  4144          4263  3                 DBG$LANGUAGE(.RSTPTR[RST$B_LANGUAGE]),
;  4145          4264  3                 DSTPTR[DST$B_MODBEG_NAME]);
;  4146          4265  3         END
;  4147          4266  2
;  4148          4267  2
;  4149          4268  2     ! If no module could be SET, modify the initialization message accordingly.
;  4150          4269  2     !
;  4151          4270  2     ELSE
;  4152          4271  3         BEGIN
;  4153          4272  3         SIGNAL(DBG$_INITIAL, 2, DBG$LANGUAGE(.DBG$GB_LANGUAGE),
;  4154          4273  3                                                 UPLIT BYTE(%ASCIC 'no module'));
;  4155          4274  3         END;
;  4156          4275  2
;  4157          4276  2
;  4158          4277  2     ! If we saw a "TRANSFER$ADDRESS" DST but it specifies the same address
;  4159          4278  2     ! as TRANSFER_ADDR, set the global back to zero - we do not need to
;  4160          4279  2     ! do anything special.
;  4161          4280  2     !
;  4162          4281  2     IF .DBG$GL_TRANSFER_ADDRESS+2 EQL .TRANSFER_ADDR
;  4163          4282  2     THEN
;  4164          4283  3         DBG$GL_TRANSFER_ADDRESS = 0;
;  4165          4284  2
;  4166          4285  2
;  4167          4286  2     ! Check for the presence of the special "TRANSFER$ADDRESS" DST
;  4168          4287  2     ! which tells us that we did not come up at the true start of the
;  4169          4288  2     ! main program. If it was present, signal an informational and
;  4170          4289  2     ! set a break at the main program.
;  4171          4290  2     !
;  4172          4291  2     IF .DBG$GL_TRANSFER_ADDRESS NEQ 0
;  4173          4292  2     THEN
;  4174          4293  3         BEGIN
;  4175          4294  3         LOCAL CMD_STRING: REF VECTOR[,BYTE];
;  4176          4295  3         SIGNAL(DBG$_NOTATMAIN);
;  4177          4296  3         CMD_STRING = DBG$GET_MEMORY(10);
;  4178          4297  3         CH$MOVE(36, UPLIT (%ASCII 'SET BREAK/TEMPORARY TRANSFER$ADDRESS'), .CMD_STRING);
;  4179          4298  3         CMD_STRING[36] = 0;
;  4180          4299  3         DBG$NCIS_ADD(.CMD_STRING, 36, CIS_INPBUF, 0, 0, 0);
;  4181          4300  3         END;
;  4182          4301  2
;  4183          4302  2
;  4184          4303  2     ! The RST initialization is done.  Now return.
;  4185          4304  2     !
;  4186          4305  2     RETURN;
;  4187          4306  2
;  4188          4307  1     END;
; INFO#250                  L1:3773
; Referenced LOCAL symbol MODNAMEPTR is probably not initialized
; INFO#250                  L1:3803
; Referenced LOCAL symbol NUM_RST_ENTS is probably not initialized
; INFO#250                  L1:3803
```

```
; Referenced LOCAL symbol NUM_RST_LONGS is probably not initialized
; INFO#250                L1:3804
; Referenced LOCAL symbol NUM_SAT_ENTS is probably not initialized
; INFO#250                L1:3810
; Referenced LOCAL symbol MOD_DSTPTR is probably not initialized
; INFO#250                L1:3813
; Referenced LOCAL symbol LANGUAGE is probably not initialized
; INFO#250                L1:3847
; Referenced LOCAL symbol TMP_SAT_CHAIN is probably not initialized
; INFO#250                L1:3894
; Referenced LOCAL symbol NUM_SAT_ENTS is probably not initialized
; INFO#250                L1:3895
; Referenced LOCAL symbol NUM_RST_ENTS is probably not initialized
; INFO#250                L1:3896
; Referenced LOCAL symbol NUM_RST_LONGS is probably not initialized
; INFO#250                L1:3907
; Referenced LOCAL symbol NUM_RST_ENTS is probably not initialized
; INFO#250                L1:3908
; Referenced LOCAL symbol NUM_RST_LONGS is probably not initialized
; INFO#250                L1:3909
; Referenced LOCAL symbol NUM_SAT_ENTS is probably not initialized
; INFO#250                L1:3910
; Referenced LOCAL symbol ROUT_STACK_INDEX is probably not initialized
; INFO#250                L1:3914
; Referenced LOCAL symbol LANGUAGE is probably not initialized
; INFO#250                L1:3930
; Referenced LOCAL symbol ROUT_STACK_INDEX is probably not initialized
; INFO#250                L1:3951
; Referenced LOCAL symbol NUM_RST_ENTS is probably not initialized
; INFO#250                L1:3952
; Referenced LOCAL symbol NUM_RST_LONGS is probably not initialized
; INFO#250                L1:3953
; Referenced LOCAL symbol NUM_SAT_ENTS is probably not initialized
; INFO#250                L1:3962
; Referenced LOCAL symbol NUM_RST_ENTS is probably not initialized
; INFO#250                L1:3963
; Referenced LOCAL symbol NUM_RST_LONGS is probably not initialized
; INFO#250                L1:3964
; Referenced LOCAL symbol NUM_SAT_ENTS is probably not initialized
; INFO#250                L1:3975
; Referenced LOCAL symbol NUM_RST_ENTS is probably not initialized
; INFO#250                L1:3979
; Referenced LOCAL symbol NUM_RST_LONGS is probably not initialized
; INFO#250                L1:3980
; Referenced LOCAL symbol NUM_SAT_ENTS is probably not initialized
; INFO#250                L1:3993
; Referenced LOCAL symbol NUM_RST_ENTS is probably not initialized
; INFO#250                L1:3994
; Referenced LOCAL symbol NUM_RST_LONGS is probably not initialized
; INFO#250                L1:3995
; Referenced LOCAL symbol NUM_SAT_ENTS is probably not initialized
; INFO#250                L1:4005
; Referenced LOCAL symbol NUM_RST_ENTS is probably not initialized
; INFO#250                L1:4006
; Referenced LOCAL symbol NUM_RST_LONGS is probably not initialized
; INFO#250                L1:4015
; Referenced LOCAL symbol NUM_SAT_ENTS is probably not initialized
```

```
; INFO#250                  L1:4034
; Referenced LOCAL symbol NUM_RST_LONGS is probably not initialized
; INFO#250                  L1:4041
; Referenced LOCAL symbol NUM_RST_LONGS is probably not initialized
; INFO#250                  L1:4049
; Referenced LOCAL symbol NUM_RST_ENTS is probably not initialized
; INFO#250                  L1:4050
; Referenced LOCAL symbol NUM_RST_LONGS is probably not initialized
; INFO#250                  L1:4059
; Referenced LOCAL symbol NUM_RST_ENTS is probably not initialized
; INFO#250                  L1:4060
; Referenced LOCAL symbol NUM_RST_LONGS is probably not initialized
; INFO#250                  L1:4075
; Referenced LOCAL symbol NUM_RST_ENTS is probably not initialized
; INFO#250                  L1:4076
; Referenced LOCAL symbol NUM_RST_LONGS is probably not initialized
; INFO#250                  L1:4083
; Referenced LOCAL symbol NUM_SAT_ENTS is probably not initialized
; INFO#250                  L1:4094
; Referenced LOCAL symbol NUM_RST_ENTS is probably not initialized
; INFO#250                  L1:4095
; Referenced LOCAL symbol NUM_RST_LONGS is probably not initialized
; INFO#250                  L1:4104
; Referenced LOCAL symbol NUM_RST_ENTS is probably not initialized
; INFO#250                  L1:4105
; Referenced LOCAL symbol NUM_RST_LONGS is probably not initialized


                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                        54  53  44  03  00100 P.AAO:  .ASCII  <3>\DST\                                    ;
                                            00000000    00104 P.AAP:  .LONG   0                                          ;
                                45  52  45  48  4E  49  47  45  42  09  00108 P.AAQ:  .ASCII  <9>\BEGINHERE\                ;
                                65  6C  75  64  6F  6D  20  6E  6E  09  00112 P.AAR:  .ASCII  <9>\no module\              ;
4F  50  4D  45  54  2F  4B  41  45  52  42  20  54  45  53  00011C P.AAS:  .ASCII  \SET BREAK/TEMPORARY TRANSFER$ADDRESS\  ;
41  24  52  45  46  53  4E  41  52  54  20  59  52  41  52  00012B                                                        ;
                            53  53  45  52  44  44          00013A                                                        ;

                                                BEGINHERENAME=      P.AAQ


                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                            0FFC 00000          .ENTRY  DBG$RST_INIT, Save R2 R3,R4,R5,R6,R7,R8,R9,-;  3401
                                                                        R10,R11                                          ;
                                5E      FECC    CE  9E 00002            MOVAB   -308(SP), SP                             ;
                                00000000'   EF  D4 00007            CLRL    RST$START_ADDR                               ;  3528
                                00000000'   EF  D4 0000D            CLRL    SAT$START_ADDR                               ;  3529
                                00000000'   EF  7C 00013            CLRQ    DST$BEGIN_ADDR                               ;  3530
                                52      0C  AC  D0 00019            MOVL    IMG_HEADER, R2                               ;  3538
                                            05  13 0001D            BEQL    1$                                           ;
                                        04  A2  B5 0001F            TSTW    4(R2)                                        ;  3545
                                            0E  12 00022            BNEQ    2$                                           ;
                                00002834A   8F  DD 00024 1$:        PUSHL   #164682                                      ;  3548
                    00000000G   00          01  FB 00027            CALLS   #1, LIB$SIGNAL                               ;
                                            04  00031            RET                                                  ;  3547
```

```
                53        04    A2  3C 00032 2$:      MOVZWL  4(R2), IMG_DST_BLOCK                    ; 3558
                53              52  C0 00036          ADDL2   R2, IMG_DST_BLOCK                       ;
                54        08    A3  3C 00039          MOVZWL  8(IMG_DST_BLOCK), DSTBLK_COUNT          ; 3559
                                07  13 0003D          BEQL    3$                                      ; 3560
    00000000'   EF              01  D0 0003F          MOVL    #1, DST$BEGIN_ADDR                      ;
                02              63  D1 00046 3$:      CMPL    (IMG_DST_BLOCK), #2                     ; 3566
                                06  14 00049          BGTR    4$                                      ;
                   00000000'    EF  D4 0004B          CLRL    DST$BEGIN_ADDR                          ;
                22 00000000G    00  E9 00051 4$:      BLBC    DBG$GV_CONTROL, 5$                      ; 3574
                   00000000'    EF  D5 00058          TSTL    DST$BEGIN_ADDR                          ;
                                1A  13 0005E          BEQL    5$                                      ;
                                53  DD 00060          PUSHL   IMG_DST_BLOCK                           ; 3577
                           08   AC  DD 00062          PUSHL   FILENAME                                ;
          0000V CF              02  FB 00065          CALLS   #2, TEST_DST_SIZE                       ;
                55              50  D0 0006A          MOVL    R0, TESTDSTSIZE                         ;
                50        01FF  C5  9E 0006D          MOVAB   511(R5), R0                             ; 3578
54              50 00000200     8F  C7 00072          DIVL3   #512, R0, DSTBLK_COUNT                  ;
                   00000000'    EF  D5 0007A 5$:      TSTL    DST$BEGIN_ADDR                          ; 3585
                                08  12 00080          BNEQ    6$                                      ;
                   00028053     8F  DD 00082          PUSHL   #163923                                 ; 3587
                                53  11 00088          BRB     7$                                      ;
             F0 AD        C8    8F  9A 0008A 6$:      MOVZBL  #200, EXE_SECBOUNDS                     ; 3598
             F4 AD 40000000     8F  D0 0008F          MOVL    #1073741824, EXE_SECBOUNDS+4            ; 3599
             E8 AD              03  D0 00097          MOVL    #3, EXE_SECNAM_DESC                     ; 3600
             EC AD 00000000'    EF  9E 0009B          MOVAB   P.AAO, EXE_SECNAM_DESC+4                ; 3601
                                7E  7C 000A3          CLRQ    -(SP)                                   ; 3602
                                63  DD 000A5          PUSHL   (IMG_DST_BLOCK)                         ; 3605
                                54  DD 000A7          PUSHL   DSTBLK_COUNT                            ; 3604
                           04   AC  DD 000A9          PUSHL   IMGFILCHAN                              ;
                                7E  7C 000AC          CLRQ    -(SP)                                   ; 3602
                           E8   AD  9F 000AE          PUSHAB  EXE_SECNAM_DESC                         ;
                   00020000     8F  DD 000B1          PUSHL   #131072                                 ;
                                7E  D4 000B7          CLRL    -(SP)                                   ;
                           F0   AD  9F 000B9          PUSHAB  EXE_SECBOUNDS                           ;
                           F0   AD  9F 000BC          PUSHAB  EXE_SECBOUNDS                           ;
          000000000G        00      0C  FB 000BF          CALLS   #12, SYS$CRMPSC                     ;
                24  AE              50  D0 000C6          MOVL    R0, STATUS                          ;
                18        24  AE    E8  000CA          BLBS    STATUS, 8$                              ; 3606
                   00000000'    EF  D4 000CE          CLRL    DST$BEGIN_ADDR                          ; 3609
24  AE          03              00  03  F0 000D4          INSV    #3, #0, #3, STATUS                  ; 3610
                           24   AE  DD 000DA          PUSHL   STATUS                                  ; 3611
          000000000G        00      01  FB 000DD 7$:      CALLS   #1, LIB$SIGNAL                      ;
                                2A  11 000E4          BRB     10$                                     ; 3606
          00000000'   EF        F0  AD  7D 000E6 8$:      MOVQ    EXE_SECBOUNDS, DST$BEGIN_ADDR       ; 3616
                   10 000000000G    00  E9 000EE          BLBC    DBG$GV_CONTROL, 9$                  ; 3618
50                 55 00000000'    EF  C1 000F5          ADDL3   DST$BEGIN_ADDR, TESTDSTSIZE, R0     ; 3620
          00000000'   EF        FF  A0  9E 000FD          MOVAB   -1(R0), DST$END_ADDR                ;
          00000000'   EF 00000000'  EF  D0 00105 9$:      MOVL    DST$BEGIN_ADDR, DST$NEXT_ADDR       ; 3622
                                0C  DD 00110 10$:     PUSHL   #12                                     ; 3647
             0000V CF               01  FB 00112          CALLS   #1, GET_MEM                         ;
               38  AE               50  D0 00117          MOVL    R0, RSTPTR                          ;
               14  A0               01  90 0011B          MOVB    #1, 20(R0)                          ; 3648
               28  A0               04  88 0011F          BISB2   #4, 40(R0)                          ; 3649
                           29   A0    94  00123          CLRB    41(R0)                              ; 3650
          00000000'   EF            50  D0 00126          MOVL    R0, RST$START_ADDR                  ; 3655
          00000000'   EF            50  D0 0012D          MOVL    R0, PREV_RST_PTR                    ; 3656
                                16  DD 00134          PUSHL   #22                                     ; 3664
```

```
                0000V  CF              01  FB 00136          CALLS   #1, GET_MEM                                    ;
            00000000'  EF              50  D0 0013B          MOVL    R0, RST$REF_LIST                               ;
                       60              14  7D 00142          MOVQ    #20, (R0)                                      ; 3665
                                       03  DD 00145          PUSHL   #3                                             ; 3673
                0000V  CF              01  FB 00147          CALLS   #1, GET_MEM                                    ;
            00000000'  EF              50  D0 0014C          MOVL    R0, LRU$LISTHEAD                               ;
                       60              50  D0 00153          MOVL    R0, (R0)                                       ; 3674
                    04 A0              50  D0 00156          MOVL    R0, 4(R0)                                      ; 3675
                             08        A0  D4 0015A          CLRL    8(R0)                                          ; 3676
            00000000G  00              00  FB 0015D          CALLS   #0, DBG$SRC_INIT                               ; 3682
            00000000G  00              00  FB 00164          CALLS   #0, DBG$SCR_INITIALIZE                         ; 3687
                                       50  D4 0016B          CLRL    I                                              ; 3694
                             D0 AD40   D4  D4 0016D 11$:     CLRL    MOD_SIZE_TBL[I]                                ; 3695
            F8                 50      05  F3 00171          AOBLEQ  #5, I, 1T$                                     ;
                             10        AE  D4 00175          CLRL    IN_MODULE_FLAG                                 ; 3697
                             18        AE  D4 00178          CLRL    ERROR_COUNT                                    ; 3698
                             08        AE  D4 0017B          CLRL    REC_NESTING                                    ; 3699
                      00000000'        EF  D4 0017E          CLRL    SAT$START_ADDR                                 ; 3700
                      57 00000000'     EF  D0 00184          MOVL    DST$BEGIN_ADDR, DSTPTR                         ; 3709
            22           20  A2            05  E1 0018B          BBC     #5, 32(R2), 12$                            ; 3710
                             04        03  A7  D1 00190          CMPL    3(DSTPTR), #4                              ; 3711
                                       1C  13 00194          BEQL    12$                                            ;
                             D0        AD  9F 00196          PUSHAB  MOD_SIZE_TBL                                   ; 3715
                                       53  DD 00199          PUSHL   IMG_DST_BLOCK                                  ;
                             04        AC  DD 0019B          PUSHL   IMGFILCHAN                                     ; 3714
                          0000V  CF        03  FB 0019E          CALLS   #3, READ_DEBUG_MODULE_TABLE                    ;
                            24   AE        50  D0 001A3          MOVL    R0, STATUS                                     ;
                                 07  24    AE  E9 001A7          BLBC    STATUS, 12$                                    ; 3716
                                 57 00000000' EF  D0 001AB          MOVL    DST$END_ADDR, DSTPTR                       ;
                00000000'        EF              57  D1 001B2 12$:     CMPL    DSTPTR, DST$END_ADDR                   ; 3727
                                                 03  1F 001B9          BLSSU   14$                                    ;
                                              0480   31 001BB 13$:     BRW     63$                                    ;
                                                67   95 001BE 14$:     TSTB    (DSTPTR)                               ; 3734
                                                07   12 001C0          BNEQ    15$                                    ;
                                           6E        0101  8F  3C 001C2          MOVZWL  #257, TYPE                     ; 3736
                                                04   11 001C7          BRB     16$                                    ;
                                           6E          01  A7  9A 001C9 15$:     MOVZBL  1(DSTPTR), TYPE                ; 3738
                                           09          10  AE  E8 001CD 16$:     BLBS    IN_MODULE_FLAG, 17$            ; 3739
                                  000000BC     8F          6E  D1 001D1          CMPL    TYPE, #188                     ;
                                                           E1  12 001D8          BNEQ    13$                            ;
                    00000101        8F                     00  6E CF 001DA 17$:     CASEL   TYPE, #0, #257         ; 3740
            0416            0416            0416                   0416     001E2 18$:     .WORD   57$-18$,-              ;
            0416            0416            0416                   0416     001EA          57$-18$,-              ;
            0416            0416            0416                   0416     001F2          57$-18$,-              ;
            0416            0416            0416                   0416     001FA          57$-18$,-              ;
            0416            0416            0416                   0416     00202          57$-18$,-              ;
            0416            0416            0416                   0416     0020A          57$-18$,-              ;
            0416            0416            0416                   0416     00212          57$-18$,-              ;
            0416            0416            0416                   0416     0021A          57$-18$,-              ;
            0416            0416            0416                   0416     00222          57$-18$,-              ;
            0451            0451            0416                   0416     0022A          57$-18$,-              ;
            0451            0451            0451                   0451     00232          57$-18$,-              ;
            0451            0451            0451                   0451     0023A          57$-18$,-              ;
            0451            0451            0451                   0451     00242          57$-18$,-              ;
            0451            0451            0451                   0451     0024A          57$-18$,-              ;
            0451            0451            0451                   0451     00252          57$-18$,-              ;
            0451            0451            0451                   0451     0025A          57$-18$,-              ;
```

0451	0451	0451	0451	00262	57-18,-
0451	0451	0451	0451	0026A	57-18,-
0451	0451	0451	0451	00272	57-18,-
0451	0451	0451	0451	0027A	57-18,-
0451	0451	0451	0451	00282	57-18,-
0451	0451	0451	0451	0028A	57-18,-
0451	0451	0451	0451	00292	57-18,-
0451	0451	0451	0451	0029A	57-18,-
0451	0451	0451	0451	002A2	57-18,-
0451	0451	0451	0451	002AA	57-18,-
0451	0451	0451	0451	002B2	57-18,-
0451	0451	0451	0451	002BA	57-18,-
0451	0451	0451	0451	002C2	57-18,-
0451	0451	0451	0451	002CA	57-18,-
0451	0451	0451	0451	002D2	57-18,-
0451	0451	0451	0451	002DA	57-18,-
0451	0451	0451	0451	002E2	57-18,-
0451	0451	0451	0451	002EA	57-18,-
0451	0451	0451	0451	002F2	57-18,-
0451	0451	0451	0451	002FA	57-18,-
0451	0451	0451	0451	00302	57-18,-
0451	0451	0451	0451	0030A	57-18,-
0451	0451	0451	0451	00312	62-18,-
0451	0416	0403	0451	0031A	62-18,-
0416	0451	0451	0451	00322	62-18,-
044E	0451	0408	040F	0032A	62-18,-
03D6	0451	0451	0451	00332	62-18,-
0451	0451	0451	03E8	0033A	62-18,-
0451	044C	0451	03B4	00342	62-18,-
044C	0451	03CF	0451	0034A	62-18,-
03BB	03BF	0451	0321	00352	62-18,-
037A	034C	026F	023E	0035A	62-18,-
0451	0451	0451	0451	00362	62-18,-
0451	0451	0451	0451	0036A	62-18,-
0451	0451	0451	0451	00372	62-18,-
0451	0451	0451	0451	0037A	62-18,-
0451	0451	0451	0451	00382	62-18,-
0451	0451	0451	0451	0038A	62-18,-
0451	0451	0451	0451	00392	62-18,-
0451	0451	0451	0451	0039A	62-18,-
0451	0451	0451	0451	003A2	62-18,-
0451	0451	0451	0451	003AA	62-18,-
0451	0451	0451	0451	003B2	62-18,-
0451	0451	0451	0451	003BA	62-18,-
0451	0451	0451	0451	003C2	62-18,-
0451	0451	0451	0451	003CA	62-18,-
0451	0451	0451	0451	003D2	62-18,-
0451	0451	0451	0451	003DA	62-18,-
		0204	0213	003E2	62-18,-

```
625-18$,-
625-18$,-
625-18$,-
```

```
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
52$-18$,-
57$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
57$-18$,-
55$-18$,-
53$-18$,-
62$-18$,-
61$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
48$-18$,-
50$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
42$-18$,-
62$-18$,-
60$-18$,-
62$-18$,-
62$-18$,-
62$-18$,-
46$-18$,-
62$-18$,-
60$-18$,-
35$-18$,-
62$-18$,-
44$-18$,-
```

```
43$-18$,-
23$-18$,-
26$-18$,-
36$-18$,-
39$-18$,-
62$-18$,-
62$-18$,-
62
```

```
                                                                62$-18$,-
                                                                62$-18$,-
                                                                62$-18$,-
                                                                62$-18$,-
                                                                62$-18$,-
                                                                62$-18$,-
                                                                62$-18$,-
                                                                62$-18$,-
                                                                62$-18$,-
                                                                62$-18$,-
                                                                62$-18$,-
                                                                62$-18$,-
                                                                20$-18$,-
                                                                19$-18$
                0002875B   8F   DD 003E6 19$:    PUSHL   #165723                                  ;3750
00000000G   00             01   FB 003EC        CALLS   #1, LIB$SIGNAL
                           59   11 003F3        BRB     25$                                      ;3740
                32      18 AE   D1 003F5 20$:    CMPL    ERROR_COUNT, #50                         ;3760
                           0D   18 003F9        BGEQ    21$
                00028643   8F   DD 003FB        PUSHL   #165443
00000000G   00             01   FB 00401        CALLS   #1, LIB$SIGNAL
                32      18 AE   D1 00408 21$:    CMPL    ERROR_COUNT, #50                         ;3761
                           0D   12 0040C        BNEQ    22$
                00028653   8F   DD 0040E        PUSHL   #165459
00000000G   00             01   FB 00414        CALLS   #1, LIB$SIGNAL
                      18 AE   D6 0041B 22$:    INCL    ERROR_COUNT                              ;3762
                           2E   11 0041E        BRB     25$                                      ;3740
                11    10 AE   E9 00420 23$:    BLBC    IN_MODULE_FLAG, 24$                      ;3773
                           54   DD 00424        PUSHL   MODNAMEPTR
                           01   DD 00426        PUSHL   #1
                0002864B   8F   DD 00428        PUSHL   #165451
      00000000G   00             03   FB 0042E        CALLS   #3, LIB$SIGNAL
            10    AE             01   D0 00435 24$:    MOVL    #1, IN_MODULE_FLAG                  ;3774
                  53             57   D0 00439        MOVL    DSTPTR, MOD_DSTPTR                  ;3775
                  54      07 A3   9E 0043C        MOVAB   7(R3), MODNAMEPTR                   ;3776
                                 58   7C 00440        CLRQ    NUM_RST_LONGS                       ;3778
                                 52   D4 00442        CLRL    NUM_SAT_ENTS                        ;3779
                              34 AE   D4 00444        CLRL    TMP_SAT_CHAIN                       ;3780
                                 56   D4 00447        CLRL    ROUT_STACK_INDEX                    ;3781
            14    AE          03 A3   D0 00449        MOVL    3(MOD_DSTPTR), LANGUAGE             ;3782
                           01E2 31   0044E 25$:    BRW     62$                                      ;3740
                              10 AE   D4 00451 26$:    CLRL    IN_MODULE_FLAG                      ;3795
      51                    59   58   C1 00454        ADDL3   NUM_RST_LONGS, NUM_RST_ENTS, R1     ;3803
      50                    52   06   C5 00458        MULL3   #6, NUM_SAT_ENTS, R0                ;3804
04    AE                    51   50   C1 0045C        ADDL3   R0, R1, RSTSIZE
                                 0C   DD 00461        PUSHL   #12                                 ;3809
                  0000V      CF   01   FB 00463        CALLS   #1, GET_MEM
                  38    AE   50   D0 00468        MOVL    R0, RSTPTR
                              55   38 AE   D0 0046C        MOVL    RSTPTR, R5                      ;3810
                  0C    A5   53   D0 00470        MOVL    MOD_DSTPTR, 12(R5)
                  14    A5   01   90 00474        MOVB    #1, 20(R5)                              ;3811
20    A5          04    AE   02   78 00478        ASHL    #2, RSTSIZE, 32(R5)                     ;3812
                  29    A5      14 AE   90 0047E        MOVB    LANGUAGE, 41(R5)                ;3813
                           50 00000000' EF   D0 00483        MOVL    PREV_RST_PTR, R0            ;3818
                  10    A0   55   D0 0048A        MOVL    R5, T6(R0)
            00000000'   EF   55   D0 0048E        MOVL    R5, PREV_RST_PTR                        ;3819
                           51   D4 00495        CLRL    I                                       ;3826
```

```
                DO AD41        04    AE   D1 00497 27$:    CMPL    RSTSIZE, MOD_SIZE_TBL[I]                    ; 3828
                                     1B   15 0049D         BLEQ    30$
                     50              04   D0 0049F         MOVL    #4, J                                       ; 3831
                                     09   11 004A2         BRB     29$
                D4 AD40        D0 AD40   D0 004A4 28$:     MOVL    MOD_SIZE_TBL[J], MOD_SIZE_TBL+4[J]          ; 3832
                                     50   D7 004AB         DECL    J
                     51              50   D1 004AD 29$:     CMPL    J, I
                                     F2   18 004B0         BGEQ    28$
                DO AD41        04    AE   D0 004B2         MOVL    RSTSIZE, MOD_SIZE_TBL[I]                    ; 3834
                                     04   11 004B8         BRB     31$                                         ; 3830
          D9         51              05   F3 004BA 30$:     AOBLEQ  #5, I, 27$                                  ; 3826
                0C   AE 00000000'    EF   9E 004BE 31$:     MOVAB   SAT$START_ADDR, PREV_SAT_PTR                ; 3845
                     5B 00000000'    EF   D0 004C6         MOVL    SAT$START_ADDR, PROG_SAT_PTR                ; 3846
                     5A        34    AE   D0 004CD         MOVL    TMP_SAT_CHAIN, TMP_SAT_PTR                  ; 3847
                                     03   12 004D1 32$:     BNEQ    33$                                         ; 3848
                                 015D   31 004D3         BRW     62$
                                     5B   D5 004D6 33$:     TSTL    PROG_SAT_PTR                                ; 3855
                                     10   13 004D8         BEQL    34$
                04   AA        04    AB   D1 004DA         CMPL    4(PROG_SAT_PTR), 4(TMP_SAT_PTR)             ; 3857
                                     09   1A 004DF         BGTRU   34$
                0C   AE              5B   D0 004E1         MOVL    PROG_SAT_PTR, PREV_SAT_PTR                  ; 3861
                     5B              6B   D0 004E5         MOVL    (PROG_SAT_PTR), PROG_SAT_PTR                ; 3862
                                     EC   11 004E8         BRB     33$                                         ; 3855
                20   AE              6A   D0 004EA 34$:     MOVL    (TMP_SAT_PTR), NEXTSAT                      ; 3870
                0C   AA              55   D0 004EE         MOVL    R5, T2(TMP_SAT_PTR)                         ; 3871
                     6A              5B   D0 004F2         MOVL    PROG_SAT_PTR, T(TMP_SAT_PTR)                ; 3872
                0C   BE              5A   D0 004F5         MOVL    TMP_SAT_PTR, @PREV_SAT_PTR                  ; 3873
                0C   AE              5A   D0 004F9         MOVL    TMP_SAT_PTR, PREV_SAT_PTR                   ; 3874
                     5A        20    AE   D0 004FD         MOVL    NEXTSAT, TMP_SAT_PTR                        ; 3875
                                     CE   11 00501         BRB     32$                                         ; 3848
                30   AE        03    A7   D0 00503 35$:     MOVL    3(DSTPTR), STARTADDR                        ; 3889
                     50        07    A7   9A 00508         MOVZBL  7(DSTPTR), R0                               ; 3891
                1C   AE        08 A047    9E 0050C         MOVAB   8(R0)[DSTPTR], PPTR
                2C   AE        1C    BE   D0 00512         MOVL    @PPTR, LENGTH                               ; 3892
                                     01   DD 00517         PUSHL   #1                                          ; 3893
                                     38   AE 9F 00519         PUSHAB  TMP_SAT_CHAIN
                                     7E   D4 0051C         CLRL    -(SP)
                                     38   AE DD 0051E         PUSHL   LENGTH
                                     40   AE DD 00521         PUSHL   STARTADDR
              0000V  CF                   05   FB 00524         CALLS   #5, STATIC_ADDRESS
                                     52   D6 00529         INCL    NUM_SAT_ENTS                                ; 3894
                                   0100   31 0052B         BRW     60$                                         ; 3895
                                     59   D6 0052E 36$:     INCL    NUM_RST_ENTS                                ; 3907
                     58              0B   C0 00530         ADDL2   #11, NUM_RST_LONGS                          ; 3908
                                     52   D6 00533         INCL    NUM_SAT_ENTS                                ; 3909
                     32              56   D1 00535         CMPL    ROUT_STACK_INDEX, #50                       ; 3910
                                     08   18 00538         BGEQ    37$
                3C AE46        03    A7   D0 0053A         MOVL    3(DSTPTR), ROUT_STACK[ROUT_STACK_INDEX]     ; 3912
                                     16   11 00540         BRB     38$
                     14        AE   D5 00542 37$:     TSTL    LANGUAGE                                    ; 3914
                                     11   13 00545         BEQL    38$
                                     54   DD 00547         PUSHL   MODNAMEPTR                                  ; 3916
                                     01   DD 00549         PUSHL   #1
                          00028962   8F   DD 0054B         PUSHL   #166242
          00000000G  00              03   FB 00551         CALLS   #3, LIB$SIGNAL
                                     56   D6 00558 38$:     INCL    ROUT_STACK_INDEX                            ; 3918
                                     38   11 0055A         BRB     41$                                         ; 3740
```

```
                13              56  F4 0055C 39$:    SOBGEQ  ROUT_STACK_INDEX, 40$                          ; 3930
                                54  DD 0055F         PUSHL   MODNAMEPTR                                     ; 3933
                                01  DD 00561         PUSHL   #1                                             ;
                    0002896A    8F  DD 00563         PUSHL   #166250                                        ;
00000000G       00              03  FB 00569         CALLS   #3, LIB$SIGNAL                                 ;
                                7D  11 00570         BRB     54$                                            ;
                32              56  D1 00572 40$:    CMPL    ROUT_STACK_INDEX, #50                          ; 3935
                                7F  18 00575         BGEQ    56$                                            ;
             30 AE          3C AE46  D0 00577         MOVL    ROUT_STACK[ROUT_STACK_INDEX], STARTADDR        ; 3938
             2C AE          03  A7  D0 0057D         MOVL    3(DSTPTR), LENGTH                              ; 3939
                                01  DD 00582         PUSHL   #1                                             ; 3940
                            38  AE  9F 00584         PUSHAB  TMP_SAT_CHAIN                                  ;
                                7E  D4 00587         CLRL    -(SP)                                          ;
                            38  AE  DD 00589         PUSHL   LENGTH                                         ;
                            40  AE  DD 0058C         PUSHL   STARTADDR                                      ;
          0000V CF                  05  FB 0058F         CALLS   #5, STATIC_ADDRESS                             ;
                                60  11 00594 41$:    BRB     56$                                            ; 3740
                                59  D6 00596 42$:    INCL    NUM_RST_ENTS                                   ; 3951
                58              08  C0 00598         ADDL2   #8, NUM_RST_LONGS                              ; 3952
                                19  11 0059B         BRB     47$                                            ; 3953
                                59  D6 0059D 43$:    INCL    NUM_RST_ENTS                                   ; 3962
                                0B  11 0059F         BRB     45$                                            ; 3963
                                59  D6 005A1 44$:    INCL    NUM_RST_ENTS                                   ; 3975
                01          02  A7  91 005A3         CMPB    2(DSTPTR), #1                                  ; 3976
                                03  13 005A7         BEQL    45$                                            ;
                              0084  31 005A9         BRW     61$                                            ;
                58              07  C0 005AC 45$:    ADDL2   #7, NUM_RST_LONGS                              ; 3979
                                79  11 005AF         BRB     59$                                            ; 3980
                                59  D6 005B1 46$:    INCL    NUM_RST_ENTS                                   ; 3993
                58              0B  C0 005B3         ADDL2   #11, NUM_RST_LONGS                             ; 3994
                                72  11 005B6 47$:    BRB     59$                                            ; 3995
                59              02  C0 005B8 48$:    ADDL2   #2, NUM_RST_ENTS                               ; 4005
                58              13  C0 005BB         ADDL2   #19, NUM_RST_LONGS                             ; 4007
                            08  AE  D5 005BE         TSTL    REC_NESTING                                    ; 4008
                                02  15 005C1         BLEQ    49$                                            ;
                                58  D6 005C3         INCL    NUM_RST_LONGS                                  ;
                            08  AE  D6 005C5 49$:    INCL    REC_NESTING                                    ; 4009
                                3A  11 005C8         BRB     58$                                            ; 4010
                            08  AE  D5 005CA 50$:    TSTL    REC_NESTING                                    ; 4025
                                11  14 005CD         BGTR    51$                                            ;
                                54  DD 005CF         PUSHL   MODNAMEPTR                                     ;
                                01  DD 005D1         PUSHL   #1                                             ;
                    0002896A    8F  DD 005D3         PUSHL   #166250                                        ;
00000000G       00              03  FB 005D9         CALLS   #3, LIB$SIGNAL                                 ;
                            08  AE  D7 005E0 51$:    DECL    REC_NESTING                                    ; 4026
                                4E  11 005E3         BRB     62$                                            ; 3740
                58              03  C0 005E5 52$:    ADDL2   #3, NUM_RST_LONGS                              ; 4041
                                49  11 005E8         BRB     62$                                            ;
                                59  D6 005EA 53$:    INCL    NUM_RST_ENTS                                   ; 4049
                58              0B  C0 005EC         ADDL2   #11, NUM_RST_LONGS                             ; 4050
                                42  11 005EF 54$:    BRB     62$                                            ; 3740
                                59  D6 005F1 55$:    INCL    NUM_RST_ENTS                                   ; 4059
                58              08  C0 005F3         ADDL2   #8, NUM_RST_LONGS                              ; 4060
                                3B  11 005F6 56$:    BRB     62$                                            ; 3740
                                59  D6 005F8 57$:    INCL    NUM_RST_ENTS                                   ; 4075
                58              07  C0 005FA         ADDL2   #7, NUM_RST_LONGS                              ; 4076
                            08  AE  D5 005FD         TSTL    REC_NESTING                                    ; 4077
```

```
                                 02  15 00600          BLEQ    58$                                     ;
                                 58  D6 00602          INCL    NUM_RST_LONGS                           ;
                             28  AE  9F 00604 58$:      PUSHAB  STATIC_FLAG                             ; 4078
                             30  AE  9F 00607          PUSHAB  LENGTH                                  ;
                             38  AE  9F 0060A          PUSHAB  STARTADDR                               ;
                                 57  DD 0060D          PUSHL   DSTPTR                                  ;
                      0000V  CF      04  FB 0060F      CALLS   #4, STATIC_CHECK                        ;
                             1B      28  AE  E9 00614  BLBC    STATIC_FLAG, 62$                        ; 4079
                                     01  DD 00618      PUSHL   #1                                      ; 4082
                             38      AE  9F 0061A      PUSHAB  TMP_SAT_CHAIN                           ;
                                     7E  D4 0061D      CLRL    -(SP)                                   ;
                             38      AE  DD 0061F      PUSHL   LENGTH                                  ;
                             40      AE  DD 00622      PUSHL   STARTADDR                               ;
                      0000V  CF          05  FB 00625  CALLS   #5, STATIC_ADDRESS                      ;
                                         52  D6 0062A 59$:     INCL    NUM_SAT_ENTS                    ; 4083
                                         05  11 0062C  BRB     62$                                     ; 3740
                                         59  D6 0062E 60$:     INCL    NUM_RST_ENTS                    ; 4104
                             58          07  C0 00630 61$:     ADDL2   #7, NUM_RST_LONGS               ; 4105
                             50          67  9A 00633 62$:     MOVZBL  (DSTPTR), R0                    ; 4119
                             57      01  A047  9E 00636       MOVAB   1(R0)[DSTPTR], DSTPTR           ;
                                     FB74  31 00639          BRW     12$                             ; 3727
                             11      10  AE  E9 0063E 63$:     BLBC    IN_MODULE_FLAG, 64$             ; 4127
                                         54  DD 00642         PUSHL   MODNAMEPTR                      ; 4129
                                         01  DD 00644         PUSHL   #1                              ;
                                 0002864B  8F  DD 00646       PUSHL   #165451                         ;
                  00000000G      00          03  FB 0064C   CALLS   #3, LIB$SIGNAL                   ;
                                             50  D4 00653 64$:     CLRL    I                           ; 4135
                51                 50          01  78 00655 65$:     ASHL    #1, I, R1               ; 4137
                                   53 00000000'EF41  DE 00659       MOVAL   RST$HASH_TABLE[R1], R3          ;
                                   63              53  D0 00661   MOVL    R3, (R3)                        ;
                00000000'EF        41              53  D0 00664   MOVL    R3, RST$HASH_TABLE+4[R1]        ; 4138
                E1                 50 000009C6      8F  F3 0066C   AOBLEQ  #2502, I, 65$                  ; 4135
                         38        AE 00000000'     EF  D0 00674   MOVL    RST$START_ADDR, RSTPTR          ; 4141
                                   50          38   AE  D0 0067C   MOVL    RSTPTR, R0                      ; 4142
                         38        AE          10   A0  D0 00680   MOVL    16(R0), RSTPTR                  ;
                                   53          38   AE  D0 00685 66$:     MOVL    RSTPTR, R3              ; 4143
                                               0E   13 00689      BEQL    67$                             ;
                                               53   DD 0068B      PUSHL   R3                              ; 4145
                               E53D    CF              01  FB 0068D   CALLS   #1, DBG$HASH_INSERT          ;
                                 38    AE          10  A3  D0 00692   MOVL    16(R3), RSTPTR              ; 4146
                                               EC   11 00697      BRB     66$                             ; 4143
                0A 00000000G   00              04  E1  00699 67$:     BBC     #4, DBG$GV_CONTROL+1, 68$   ; 4158
                                       04      AC  DD 006A1       PUSHL   IMGFILCHAN                      ; 4160
                               0000V   CF              01  FB 006A4   CALLS   #1, SHARED_IMAGE_NAMES      ;
                                               08   11 006A9      BRB     69$                             ;
                                       04      AC  DD 006AB 68$:     PUSHL   IMGFILCHAN                  ; 4162
                               0000V   CF              01  FB 006AE   CALLS   #1, SHARED_IMAGE_NAMES_V3   ;
                                           00000000'   EF  DD 006B3 69$:     PUSHL   RST$START_ADDR      ; 4167
                                                       7E  D4 006B9       CLRL    -(SP)                   ;
                                       04      AC  DD 006BB       PUSHL   IMGFILCHAN                      ;
                                E098   CF              03  FB 006BE   CALLS   #3, DBG$GST_INIT          ;
                                       0D              50  E8 006C3   BLBS    R0, 70$                     ;
                                           000280A3    8F  DD 006C6   PUSHL   #164003                     ; 4169
                                   00000000G   00          01  FB 006CC   CALLS   #1, LIB$SIGNAL         ;
                                               54      2000    8F  3C 006D3 70$:     MOVZWL  #8192, LONGS_NEEDED ; 4180
                                                               50  D4 006D8       CLRL    I               ; 4181
                                               54          D0 AD40  C0 006DA 71$:     ADDL2   MOD_SIZE_TBL[I], LONGS_NEEDED ; 4182
```

```
                F7              50              05  F3 006DF           AOBLEQ  #5, I, 71$                                     ;
                00000000G       00              00  FB 006E3           CALLS   #0, DBG$FREE_MEM_LEFT                          ; 4184
                                54              50  C2 006EA           SUBL2   R0, LONGS_NEEDED
                                53              06  D0 006ED           MOVL    #6, MST_INDEX                                  ; 4185
                00000064        8F                  54  D1 006F0 72$:  CMPL    LONGS_NEEDED, #100                             ; 4186
                                                    1E  15 006F7           BLEQ    73$
                                                    54  DD 006F9           PUSHL   LONGS_NEEDED                           ; 4188
                00000000G       00                  01  FB 006FB           CALLS   #1, DBG$EXPAND_MEMORY
                          24    AE                  50  D0 00702           MOVL    R0, STATUS
                                0D          24      AE  E8 00706           BLBS    STATUS, 73$                            ; 4189
                                                    53  D5 0070A           TSTL    MST_INDEX
                                                    09  13 0070C           BEQL    73$
                                                    53  D7 0070E           DECL    MST_INDEX                              ; 4190
                                54          D0 AD43  C2 00710           SUBL2   MOD_SIZE_TBL[MST_INDEX], LONGS_NEEDED  ; 4191
                                                    D9  11 00715           BRB     72$                                    ; 4186
                                            38      AE  9F 00717 73$:      PUSHAB  RSTPTR                                 ; 4200
                                00000000'           EF  9F 0071A           PUSHAB  P.AAP
                          0000V CF                      02  FB 00720           CALLS   #2, DBG$RST_SETSCOPE
                                            38      AE  D4 00725           CLRL    RSTPTR                                 ; 4206
                                50 00000000'        EF  D0 00728           MOVL    SAT$START_ADDR, SATPTR                 ; 4207
                                                    1A  13 0072F 74$:      BEQL    76$                                    ; 4208
                             04 A0              10      AC  D1 00731           CMPL    TRANSFER_ADDR, 4(SATPTR)               ; 4210
                                                    0E  19 00736           BLSS    75$
                             08 A0              10      AC  D1 00738           CMPL    TRANSFER_ADDR, 8(SATPTR)               ; 4211
                                                    07  14 0073D           BGTR    75$
                             38 AE              0C      A0  D0 0073F           MOVL    12(SATPTR), RSTPTR                     ; 4214
                                                    05  11 00744           BRB     76$                                    ; 4213
                                50                      60  D0 00746 75$:      MOVL    (SATPTR), SATPTR                       ; 4218
                                                    E4  11 00749           BRB     74$                                    ; 4208
                                            38      AE  D5 0074B 76$:      TSTL    RSTPTR                                 ; 4225
                                                    11  12 0074E           BNEQ    77$
                             38 AE      00000000'       EF  D0 00750           MOVL    RST$START_ADDR, RSTPTR                 ; 4228
                                50              38      AE  D0 00758           MOVL    RSTPTR, R0                             ; 4229
                             38 AE              10      A0  D0 0075C           MOVL    16(R0), RSTPTR
                47 00000000G    00                      01  E1 00761 77$:      BBC     #1, DBG$GV_CONTROL, 79$                ; 4237
                                00000000'           EF  9F 00769           PUSHAB  BEGINHERENAME                          ; 4241
                   00000000G    00                      01  FB 0076F           CALLS   #1, DBG$STA_LOOKUP_GBL
                         38     AE                      50  D0 00776           MOVL    R0, RSTPTR
                                52          38      AE  D0 0077A           MOVL    RSTPTR, R2                             ; 4242
                                                    15  12 0077E           BNEQ    78$
                                00000000'           EF  9F 00780           PUSHAB  BEGINHERENAME
                                                    01  DD 00786           PUSHL   #1
                                000281F8            8F  DD 00788           PUSHL   #164344
                   00000000G    00                      03  FB 0078E           CALLS   #3, LIB$SIGNAL
                                57          0C      A2  D0 00795 78$:      MOVL    12(R2), DSTPTR                         ; 4243
00000000G       00          03  A7                      02  C1 00799           ADDL3   #2, 3(DSTPTR), DBG$RUNFRAME+64         ; 4244
                                                    02  DD 007A2           PUSHL   #2                                     ; 4245
                                                    7E  D4 007A4           CLRL    -(SP)
                   00000000G    00                      02  FB 007A6           CALLS   #2, DBG$SET_LANG
                                            38      AE  D4 007AD           CLRL    RSTPTR                                 ; 4246
                                52          38      AE  D0 007B0 79$:      MOVL    RSTPTR, R2                             ; 4255
                                                    27  13 007B4           BEQL    80$
                             28 A2                      01  88 007B6           BISB2   #1, 40(R2)                             ; 4258
                                                    01  DD 007BA           PUSHL   #1                                     ; 4259
                                                    52  DD 007BC           PUSHL   R2
                          E48E  CF                      02  FB 007BE           CALLS   #2, DBG$RST_BUILD
                                7E          29      A2  9A 007C3           MOVZBL  41(R2), -(SP)                          ; 4260
```

```
                                 7E  D4 007C7          CLRL    -(SP)                                                   ;
                    00000000G    00  02 FB 007C9      CALLS   #2, DBG$SET_LANG                                        ;
                                 57      0C  A2 D0 007D0  MOVL    12(R2), DSTPTR                                     ; 4261
                                         07  A7 9F 007D4  PUSHAB  7(DSTPTR)                                          ; 4264
                                 7E      29  A2 9A 007D7  MOVZBL  41(R2), -(SP)                                      ; 4263
                                         0D  11 007DB     BRB     81$                                                ;
                              00000000'  EF  9F 007DD 80$: PUSHAB  P.AAR                                              ; 4273
                                 7E 00000000G    00  9A 007E3  MOVZBL  DBG$GB_LANGUAGE, -(SP)                         ; 4272
                    00000000G    00  01 FB 007EA 81$: CALLS   #1, DBG$LANGUAGE                                       ;
                                         50  DD 007F1     PUSHL   R0                                                 ;
                                         02  DD 007F3     PUSHL   #2                                                 ;
                                  0002800B   8F  DD 007F5  PUSHL   #163851                                           ;
                    00000000G    00  04 FB 007FB      CALLS   #4, LIB$SIGNAL                                          ;
                50  00000000G    00  02 C1 00802      ADDL3   #2, DBG$GL_TRANSFER_ADDRESS, R0                         ; 4281
                          10     AC  50 D1 0080A      CMPL    R0, TRANSFER_ADDR                                       ;
                                         06  12 0080E     BNEQ    82$                                                ;
                              00000000G  00  D4 00810     CLRL    DBG$GL_TRANSFER_ADDRESS                            ; 4283
                              00000000G  00  D5 00816 82$: TSTL    DBG$GL_TRANSFER_ADDRESS                            ; 4291
                                         34  13 0081C     BEQL    83$                                                ;
                              000287A3   8F  DD 0081E     PUSHL   #165795                                            ; 4295
                    00000000G    00  01 FB 00824      CALLS   #1, LIB$SIGNAL                                          ;
                                         0A  DD 0082B     PUSHL   #10                                                ; 4296
                    00000000G    00  01 FB 0082D      CALLS   #1, DBG$GET_MEMORY                                      ;
                                 56      50  D0 00834  MOVL    R0, CMD_STRING                                         ;
                50  00000000'    EF      24  28 00837  MOVC3   #36, P.AAS, (CMD_STRING)                             ; 4297
                                         24  A6 94 0083F  CLRB    36(CMD_STRING)                                     ; 4298
                                 7E          7C 00842     CLRQ    -(SP)                                              ; 4299
                                             7D 00844     MOVQ    #2, -(SP)                                          ;
                                         24  DD 00847     PUSHL   #36                                                ;
                                         56  DD 00849     PUSHL   CMD_STRING                                         ;
                    00000000G    00  06 FB 0084B      CALLS   #6, DBG$NCIS_ADD                                        ;
                                             04 00852 83$: RET                                                       ; 4307

; Routine Size:  2131 bytes,    Routine Base:  DBG$CODE + 1B6D
```

```
;  4190          4308  1 GLOBAL ROUTINE DBG$RST_MOST_RECENT(MODRSTPTR): NOVALUE =
;  4191          4309  1
;  4192          4310  1 !++
;  4193          4311  1 ! FUNCTION
;  4194          4312  1 !       This routine marks a specified module as being the Most Recently Used
;  4195          4313  1 !       module among the modules currently in the Run-Time Symbol Table (RST).
;  4196          4314  1 !       This is done by searching for the module in the LRUM (Least Recently
;  4197          4315  1 !       Used Module) table and moving its LRUM entry to the first position in
;  4198          4316  1 !       the table.  If no LRUM entry exists for the specified module, a new
;  4199          4317  1 !       entry is created and inserted at the beginning of the table.
;  4200          4318  1 !
;  4201          4319  1 !       The LRUM table is a doubly linked list of entries, where each entry
;  4202          4320  1 !       represents one module currently in the RST.  The list is ordered by
;  4203          4321  1 !       recency of reference, so that the first module on the list is the most
;  4204          4322  1 !       recently referenced and the last is the Least Recently Used.  This
;  4205          4323  1 !       routines moves entries to the beginning of the list--this keeps used
;  4206          4324  1 !       modules away from the Least Recently Used slot at the end of the list.
;  4207          4325  1 !
;  4208          4326  1 !       When more memory is needed in the free memory pool, DBG$GET_MEMORY will
;  4209          4327  1 !       use the LRUM table to find the Least Recently Used Module.  That module
;  4210          4328  1 !       is then removed from the RST so that its RST space can be reused.  The
;  4211          4329  1 !       Most Recently Used module will never be removed, however; this prevents
;  4212          4330  1 !       a module from being removed while its own RST is being built.
;  4213          4331  1 !
;  4214          4332  1 ! INPUTS
;  4215          4333  1 !       MODRSTPTR - A pointer to the Module RST Entry for the module to be
;  4216          4334  1 !                   marked as being Most Recently Used.
;  4217          4335  1 !
;  4218          4336  1 ! OUTPUTS
;  4219          4337  1 !       NONE
;  4220          4338  1 !
;  4221          4339  2 !--
;  4222          4340  2     BEGIN
;  4223          4341  2
;  4224          4342  2     MAP
;  4225          4343  2         MODRSTPTR: REF RST$ENTRY;       ! Pointer to Module RST Entry of the
;  4226          4344  2                                         !      Most Recently Used module
;  4227          4345  2
;  4228          4346  2     LOCAL
;  4229          4347  2         BACKPTR: REF LRUM$ENTRY,        ! Pointer to previous entry on LRUM list
;  4230          4348  2         FORWPTR: REF LRUM$ENTRY,        ! Pointer to the next entry on LRUM list
;  4231          4349  2         PTR: REF LRUM$ENTRY;            ! Pointer to the current LRUM entry
;  4232          4350  2
;  4233          4351  2
;  4234          4352  2     ! If the specified module is already marked as being the Most Recently Used
;  4235          4353  2     ! module, return right away.
;  4236          4354  2     !
;  4237          4355  2     IF .MODRSTPTR EQL .LRUM$MOST_RECENT THEN RETURN;
;  4238          4356  2
;  4239          4357  2
;  4240          4358  2     ! Search the whole Least Recently Used Module list for the MODRSTPTR module.
;  4241          4359  2     ! Exit the loop with PTR pointing to the corresponding LRUM entry.
;  4242          4360  2     !
;  4243          4361  2     PTR = .LRUM$LISTHEAD[LRUM$L_FLINK];
;  4244          4362  2     WHILE TRUE DO
;  4245          4363  3         BEGIN
;  4246          4364  3
```

```
; 4247          4365  3
; 4248          4366  3          ! If we have come to the end of the list, the specified module is not on
; 4249          4367  3          ! the LRUM list.  If so, create a new LRUM entry for the MODRSTPTR mod-
; 4250          4368  3          ! ule and exit the search loop.
; 4251          4369  3          !
; 4252          4370  3          IF .PTR EQL .LRUM$LISTHEAD
; 4253          4371  3          THEN
; 4254          4372  4              BEGIN
; 4255          4373  4              IF NOT .MODRSTPTR[RST$V_MOD_IN_RST]
; 4256          4374  4              THEN
; 4257          4375  4                  $DBG_ERROR('RSTCNTRL\RST_MOST_RECENT');
; 4258          4376  4
; 4259          4377  4              PTR = DBG$GET_MEMORY(LRUM$ENTSIZE);
; 4260          4378  4              PTR[LRUM$L_RSTPTR] = .MODRSTPTR;
; 4261          4379  4              EXITLOOP;
; 4262          4380  3              END;
; 4263          4381  3
; 4264          4382  3
; 4265          4383  3          ! If this is the desired LRUM entry, unlink it from the doubly linked
; 4266          4384  3          ! list and exit the loop.
; 4267          4385  3          !
; 4268          4386  3          IF .PTR[LRUM$L_RSTPTR] EQL .MODRSTPTR
; 4269          4387  3          THEN
; 4270          4388  4              BEGIN
; 4271          4389  4              FORWPTR = .PTR[LRUM$L_FLINK];
; 4272          4390  4              BACKPTR = .PTR[LRUM$L_BLINK];
; 4273          4391  4              FORWPTR[LRUM$L_BLINK] = .BACKPTR;
; 4274          4392  4              BACKPTR[LRUM$L_FLINK] = .FORWPTR;
; 4275          4393  4              EXITLOOP;
; 4276          4394  3              END;
; 4277          4395  3
; 4278          4396  3
; 4279          4397  3          ! We have not found it yet--link on and loop.
; 4280          4398  3          !
; 4281          4399  3          PTR = .PTR[LRUM$L_FLINK];
; 4282          4400  2          END;
; 4283          4401  2
; 4284          4402  2
; 4285          4403  2      ! We now have the desired LRUM entry.  Link it into the doubly linked LRUM
; 4286          4404  2      ! list in the Most Recently Used position.  Also set LRUM$MOST_RECENT to
; 4287          4405  2      ! point to the specified Module RST Entry.  Then return.
; 4288          4406  2      !
; 4289          4407  2      FORWPTR = .LRUM$LISTHEAD[LRUM$L_FLINK];
; 4290          4408  2      BACKPTR = .LRUM$LISTHEAD;
; 4291          4409  2      PTR[LRUM$L_FLINK] = .FORWPTR;
; 4292          4410  2      PTR[LRUM$L_BLINK] = .BACKPTR;
; 4293          4411  2      FORWPTR[LRUM$L_BLINK] = .PTR;
; 4294          4412  2      BACKPTR[LRUM$L_FLINK] = .PTR;
; 4295          4413  2      LRUM$MOST_RECENT = .MODRSTPTR;
; 4296          4414  2      RETURN;
; 4297          4415  2
; 4298          4416  1      END;

                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0
```

```
4D 5F 54 53 52 5C 4C 52 54 4E 43 54 53 52 18  00140 P.AAT:  .ASCII  <24>\RSTCNTRL\<92>\RST_MOST_RECENT\
                  54 4E 45 43 45 52 5F 54 53 4F  0014F

                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                 007C 00000         .ENTRY  DBG$RST_MOST_RECENT, Save R2,R3,R4,R5,R6    ; 4308
                                              56 00000000'  EF   9E 00002         MOVAB   LRUM$LISTHEAD, R6
                                              55       04   AC   D0 00009         MOVL    MODRSTPTR, R5               ; 4355
                                        04    A6           55   D1 0000D         CMPL    R5, LRUM$MOST_RECENT
                                                           61   13 00011         BEQL    6$
                                              52       00   B6   D0 00013         MOVL    @LRUM$LISTHEAD, PTR         ; 4361
                                              66           52   D1 00017 1$:      CMPL    PTR, LRUM$LISTHEAD          ; 4370
                                                           2C   12 0001A         BNEQ    3$
                                     15       28    A5     01   E0 0001C         BBS     #1, 40(R5), 2$              ; 4373
                                                 00000000'  EF   9F 00021         PUSHAB  P.AAT                       ; 4375
                                                           01   DD 00027         PUSHL   #1
                                                 00028362  8F   DD 00029         PUSHL   #164706
                                  00000000G      00           03   FB 0002F         CALLS   #3, LIB$SIGNAL
                                                           03   DD 00036 2$:      PUSHL   #3                          ; 4377
                                  00000000G      00           01   FB 00038         CALLS   #1, DBG$GET_MEMORY
                                              52               50   D0 0003F         MOVL    R0, PTR
                                        08    A2               55   D0 00042         MOVL    R5, 8(PTR)                  ; 4378
                                                           17   11 00046         BRB     5$                          ; 4372
                                              55       08   A2   D1 00048 3$:      CMPL    8(PTR), R5                  ; 4386
                                                           0C   12 0004C         BNEQ    4$
                                              53               62   7D 0004E         MOVQ    (PTR), FORWPTR              ; 4389
                                        04    A3               54   D0 00051         MOVL    BACKPTR, 4(FORWPTR)         ; 4391
                                              64               53   D0 00055         MOVL    FORWPTR, (BACKPTR)          ; 4392
                                                           05   11 00058         BRB     5$                          ; 4388
                                              52               62   D0 0005A 4$:      MOVL    (PTR), PTR                  ; 4399
                                                           B8   11 0005D         BRB     1$                          ; 4362
                                              53       00   B6   D0 0005F 5$:      MOVL    @LRUM$LISTHEAD, FORWPTR     ; 4407
                                              54           66   D0 00063         MOVL    LRUM$LISTHEAD, BACKPTR      ; 4408
                                              62           53   7D 00066         MOVQ    FORWPTR, (PTR)              ; 4409
                                        04    A3           52   D0 00069         MOVL    PTR, 4(FORWPTR)             ; 4411
                                              64           52   D0 0006D         MOVL    PTR, (BACKPTR)              ; 4412
                                        04    A6           55   D0 00070         MOVL    R5, LRUM$MOST_RECENT        ; 4413
                                                           04   00074 6$:      RET                                 ; 4416

; Routine Size:  117 bytes,    Routine Base:  DBG$CODE + 23C0
```

```
;  4300          4417  1 GLOBAL ROUTINE DBG$RST_NEXT_DST(PASSED_DSTPTR) =
;  4301          4418  1 !
;  4302          4419  1 ! FUNCTION
;  4303          4420  1 !       This routine searches the Moved DST entry list for a matching
;  4304          4421  1 !       "real" DST pointer.  If it find one it returnes the address of the
;  4305          4422  1 !       next DST.  If not it returns the next DST as calculated using the
;  4306          4423  1 !       length.
;  4307          4424  1 !
;  4308          4425  1 ! INPUTS
;  4309          4426  1 !       PASSED_DSTPTR - The DST pointer to look for.  Passed by value.
;  4310          4427  1 !
;  4311          4428  1 ! OUTPUTS
;  4312          4429  1 !       Returns the "next" DST pointer.
;  4313          4430  1 !
;  4314          4431  1
;  4315          4432  2     BEGIN
;  4316          4433  2
;  4317          4434  2     LOCAL
;  4318          4435  2         DSTPTR: REF DST$RECORD,          ! ???
;  4319          4436  2         MOVED_DST_ENTRY: REF MDE$RECORD;! ???
;  4320          4437  2
;  4321          4438  2
;  4322          4439  2
;  4323          4440  2     ! Assure ourselves that it is "real".
;  4324          4441  2     !
;  4325          4442  2     DSTPTR = DBG$RST_DST_PTR(.PASSED_DSTPTR);
;  4326          4443  2     MOVED_DST_ENTRY = .DBG$GL_MOVED_DST_LIST_HEAD;
;  4327          4444  2     WHILE .MOVED_DST_ENTRY NEQA 0 DO
;  4328          4445  3         BEGIN
;  4329          4446  3
;  4330          4447  3
;  4331          4448  3         ! Check if we've found it and return the "next" one.  Make sure it
;  4332          4449  3         ! hasn't moved
;  4333          4450  3         !
;  4334          4451  3         IF .DSTPTR EQLA .MOVED_DST_ENTRY[MDE$L_REAL_DSTPTR]
;  4335          4452  3         THEN
;  4336          4453  3             RETURN DBG$RST_DST_PTR(.MOVED_DST_ENTRY[MDE$L_NEXT_DSTPTR]);
;  4337          4454  3
;  4338          4455  3
;  4339          4456  3         ! Next!
;  4340          4457  3         !
;  4341          4458  3         MOVED_DST_ENTRY = .MOVED_DST_ENTRY[MDE$L_FLINK];
;  4342          4459  3
;  4343          4460  3         END;
;  4344          4461  2
;  4345          4462  2
;  4346          4463  2     ! We didn't find anything.  Make sure it hasn't moved.
;  4347          4464  2     !
;  4348          4465  2     RETURN DBG$RST_DST_PTR(DSTPTR[DST$A_NEXT] + .DSTPTR[DST$B_LENGTH]);
;  4349          4466  2
;  4350          4467  1     END;


                    000C 00000         .ENTRY  DBG$RST_NEXT_DST, Save R2,R3                ; 4417
```

```
                        04   AC  DD 00002          PUSHL   PASSED_DSTPTR                                          ; 4442
              F70D      CF       01 FB 00005       CALLS   #1, DBG$RST_DST_PTR
                        53       50 D0 0000A       MOVL    R0, DSTPTR                                            ;
                        52 00000000G  00 D0 0000D  MOVL    DBG$GL_MOVED_DST_LIST_HEAD, MOVED_DST_ENTRY           ; 4443
                                 10 13 00014 1$:   BEQL    3$                                                    ; 4444
                08      A2       53 D1 00016       CMPL    DSTPTR, 8(MOVED_DST_ENTRY)                            ; 4451
                                  05 12 0001A      BNEQ    2$
                                10 A2 DD 0001C     PUSHL   16(MOVED_DST_ENTRY)                                   ; 4453
                                  0C 11 0001F      BRB     4$
                        52        62 D0 00021 2$:  MOVL    (MOVED_DST_ENTRY), MOVED_DST_ENTRY                    ; 4458
                                  EE 11 00024      BRB     1$                                                    ; 4444
                        50        63 9A 00026 3$:  MOVZBL  (DSTPTR), R0                                          ; 4465
                                01 A043 9F 00029   PUSHAB  1(R0)[DSTPTR]
              F6E5      CF       01 FB 0002D 4$:  CALLS   #1, DBG$RST_DST_PTR
                                  04 00032         RET                                                           ; 4467

; Routine Size:  51 bytes,    Routine Base:  DBG$CODE + 2435
```

```
;  4352          4468  1 GLOBAL ROUTINE DBG$RST_REMOVE(MODRSTPTR): NOVALUE =
;  4353          4469  1 !
;  4354          4470  1 ! FUNCTION
;  4355          4471  1 !       This routine removes the RST for one module from the Run-Time Symbol
;  4356          4472  1 !       Table (RST).  It accepts as input a pointer to the Module RST Entry of
;  4357          4473  1 !       the module to be removed.  The module's symbol chain is then scanned,
;  4358          4474  1 !       and all RST entries on it with zero reference counts are released to the
;  4359          4475  1 !       free memory pool.  The module's Static Address Table chain is also re-
;  4360          4476  1 !       leased and its Least Recently Used Module (LRUM) table entry is removed.
;  4361          4477  1 !
;  4362          4478  1 !       RST entries on the module's symbol chain which are locked (have non-zero
;  4363          4479  1 !       reference counts) cannot be released since there are pointers around
;  4364          4480  1 !       which point to them.  Such entries are therefore added to the Temporary
;  4365          4481  1 !       RST Entry List after being removed from the module's symbol chain and
;  4366          4482  1 !       from the RST Hash Table.  These RST entries are eventually released by
;  4367          4483  1 !       routine DBG$RST_TEMP_RELEASE after their reference counts have been re-
;  4368          4484  1 !       duced to zero.
;  4369          4485  1 !
;  4370          4486  1 ! INPUTS
;  4371          4487  1 !       MODRSTPTR - A pointer to the Module RST Entry of the module which is to
;  4372          4488  1 !                   be removed from the RST.
;  4373          4489  1 !
;  4374          4490  1 ! OUTPUTS
;  4375          4491  1 !       NONE
;  4376          4492  1 !
;  4377          4493  1
;  4378          4494  2     BEGIN
;  4379          4495  2
;  4380          4496  2     MAP
;  4381          4497  2         MODRSTPTR: REF RST$ENTRY;       ! Pointer to the Module RST Entry of the
;  4382          4498  2                                         !      module to be removed
;  4383          4499  2
;  4384          4500  2     LOCAL
;  4385          4501  2         BACKLRUM: REF LRUM$ENTRY,       ! Pointer to previous LRUM entry
;  4386          4502  2         BACKPTR: REF RST$ENTRY,         ! Pointer to previous RST entry on
;  4387          4503  2                                         !      the RST hash chain
;  4388          4504  2         FORWLRUM: REF LRUM$ENTRY,       ! Pointer to the next LRUM entry
;  4389          4505  2         FORWPTR: REF RST$ENTRY,         ! Pointer to the next RST entry on
;  4390          4506  2                                         !      the RST hash chain
;  4391          4507  2         LOCKNODE: DBG$LINK_NODE,        ! Node used to lock or unlock SYMIDs
;  4392          4508  2         LRUMPTR: REF LRUM$ENTRY,        ! Pointer to the current LRUM (Least Re-
;  4393          4509  2                                         !      cently Used Module) table entry
;  4394          4510  2         NEXTRST,                        ! Pointer to the next RST entry on the
;  4395          4511  2                                         !      module's symbol chain
;  4396          4512  2         NEXTSAT,                        ! Pointer to the next entry on the
;  4397          4513  2                                         !      module's SAT chain
;  4398          4514  2         RSTPTR: REF RST$ENTRY,          ! Pointer to the current RST entry on
;  4399          4515  2                                         !      the module's symbol chain
;  4400          4516  2         SATPTR: REF SAT$ENTRY;          ! Pointer to the current Static Address
;  4401          4517  2                                         !      Table entry being removed
;  4402          4518  2
;  4403          4519  2
;  4404          4520  2
;  4405          4521  2     ! Mark the specified module as no longer being in the RST.  Also increment
;  4406          4522  2     ! the reference counts of all RST entries which are referenced by the cur-
;  4407          4523  2     ! rent Debug command (the entries on the RST Reference List).
;  4408          4524  2     !
```

```
;  4409          4525  2     MODRSTPTR[RST$V_MOD_IN_RST] = FALSE;
;  4410          4526  2     LOCKNODE[DBG$L_LINK_NODE_LINK] = 0;
;  4411          4527  2     INCR I FROM 1 TO .RST$REF_LIST[1] DO
;  4412          4528  3         BEGIN
;  4413          4529  3         LOCKNODE[DBG$L_LINK_NODE_VALUE] = .RST$REF_LIST[.I + 1];
;  4414          4530  3         DBG$STA_LOCK_SYMID(LOCKNODE);
;  4415          4531  2         END;
;  4416          4532  2
;  4417          4533  2
;  4418          4534  2     ! Loop over the module's symbol chain and remove all entries on it from that
;  4419          4535  2     ! chain and from the RST Hash Table.  Release all RST entries with zero ref-
;  4420          4536  2     ! erence counts to the memory pool, and put those with non-zero reference
;  4421          4537  2     ! counts on the Temporary RST Entry List.
;  4422          4538  2     !
;  4423          4539  2     RSTPTR = .MODRSTPTR[RST$L_SYMCHNPTR];
;  4424          4540  2     MODRSTPTR[RST$L_SYMCHNPTR] = 0;
;  4425          4541  2     WHILE .RSTPTR NEQ 0 DO
;  4426          4542  3         BEGIN
;  4427          4543  3         NEXTRST = .RSTPTR[RST$L_SYMCHNPTR];
;  4428          4544  3
;  4429          4545  3
;  4430          4546  3         ! Remove the RST entry from the RST Hash Table.
;  4431          4547  3         !
;  4432          4548  3         FORWPTR = .RSTPTR[RST$L_HASH_FLINK];
;  4433          4549  3         BACKPTR = .RSTPTR[RST$L_HASH_BLINK];
;  4434          4550  3         FORWPTR[RST$L_HASH_BLINK] = .BACKPTR;
;  4435          4551  3         BACKPTR[RST$L_HASH_FLINK] = .FORWPTR;
;  4436          4552  3
;  4437          4553  3
;  4438          4554  3         ! Since the SAT is going to be removed, the pointers into the SAT
;  4439          4555  3         ! should be updated (zeroed).  This is done for every routine;
;  4440          4556  3         ! especially those that are hanging around because their reference
;  4441          4557  3         ! count is not zero.  This may effect code in DBG$STA_SETCONTEXT.
;  4442          4558  3         !
;  4443          4559  3         IF .RSTPTR[RST$B_KIND] EQL RST$K_ROUTINE
;  4444          4560  3         THEN
;  4445          4561  3             RSTPTR[RST$L_RTNSATPTR] = 0;
;  4446          4562  3
;  4447          4563  3
;  4448          4564  3         ! Release the RST entry to the memory pool or put it on the
;  4449          4565  3         ! Temporary RST Entry List.
;  4450          4566  3         !
;  4451          4567  3         IF .RSTPTR[RST$W_REFCOUNT] EQL 0
;  4452          4568  3         THEN
;  4453          4569  3             DBG$REL_MEMORY(.RSTPTR)
;  4454          4570  3
;  4455          4571  3         ELSE
;  4456          4572  4             BEGIN
;  4457          4573  4             RSTPTR[RST$L_HASH_FLINK] = .RST$TEMP_LIST;
;  4458          4574  4             RSTPTR[RST$L_HASH_BLINK] = 0;
;  4459          4575  4             RST$TEMP_LIST = .RSTPTR;
;  4460          4576  4             RSTPTR[RST$L_SYMCHNPTR] = 0;
;  4461          4577  3             END;
;  4462          4578  3
;  4463          4579  3         RSTPTR = .NEXTRST;
;  4464          4580  2         END;
;  4465          4581  2
```

```
;  4466          4582  2
;  4467          4583  2      ! Also remove all SAT entries on the module's Static Address Table chain.
;  4468          4584  2      !
;  4469          4585  2      SATPTR = .MODRSTPTR[RST$L_SAT_PTR];
;  4470          4586  2      MODRSTPTR[RST$L_SAT_PTR] = 0;
;  4471          4587  2      WHILE .SATPTR NEQ 0 DO
;  4472          4588  3          BEGIN
;  4473          4589  3          NEXTSAT = .SATPTR[SAT$L_FLINK];
;  4474          4590  3          DBG$REL_MEMORY(.SATPTR);
;  4475          4591  3          SATPTR = .NEXTSAT;
;  4476          4592  2          END;
;  4477          4593  2
;  4478          4594  2
;  4479          4595  2      ! If we have PC-Correlation and Source-Correlation DST record pointer
;  4480          4596  2      ! tables, we release them here.
;  4481          4597  2      !
;  4482          4598  2      IF .MODRSTPTR[RST$L_MODPCTBL] NEQ 0
;  4483          4599  2      THEN
;  4484          4600  2          DBG$REL_MEMORY(.MODRSTPTR[RST$L_MODPCTBL]);
;  4485          4601  2
;  4486          4602  2      MODRSTPTR[RST$L_MODPCTBL] = 0;
;  4487          4603  2      IF .MODRSTPTR[RST$L_MODSRCTBL] NEQ 0
;  4488          4604  2      THEN
;  4489          4605  2          DBG$REL_MEMORY(.MODRSTPTR[RST$L_MODSRCTBL]);
;  4490          4606  2
;  4491          4607  2      MODRSTPTR[RST$L_MODSRCTBL] = 0;
;  4492          4608  2
;  4493          4609  2
;  4494          4610  2      ! Decrement the reference counts of all RST entries referenced by the
;  4495          4611  2      ! current Debug command; we have already ensured that they will stay in
;  4496          4612  2      ! memory for now.
;  4497          4613  2      !
;  4498          4614  2      INCR I FROM 1 TO .RST$REF_LIST[1] DO
;  4499          4615  3          BEGIN
;  4500          4616  3          LOCKNODE[DBG$L_LINK_NODE_VALUE] = .RST$REF_LIST[.I + 1];
;  4501          4617  3          DBG$STA_UNLOCK_SYMID(LOCKNODE);
;  4502          4618  2          END;
;  4503          4619  2
;  4504          4620  2
;  4505          4621  2      ! Finally remove the LRUM (Least Recently Used Module) table entry for the
;  4506          4622  2      ! module.  Such entries may only exist for modules currently in the RST.
;  4507          4623  2      !
;  4508          4624  2      IF .MODRSTPTR EQL .LRUM$MOST_RECENT THEN LRUM$MOST_RECENT = 0;
;  4509          4625  2      LRUMPTR = .LRUM$LISTHEAD[LRUM$L_FLINK];
;  4510          4626  2      WHILE .LRUMPTR NEQ .LRUM$LISTHEAD DO
;  4511          4627  3          BEGIN
;  4512          4628  3          IF .LRUMPTR[LRUM$L_RSTPTR] EQL .MODRSTPTR
;  4513          4629  3          THEN
;  4514          4630  4              BEGIN
;  4515          4631  4              FORWLRUM = .LRUMPTR[LRUM$L_FLINK];
;  4516          4632  4              BACKLRUM = .LRUMPTR[LRUM$L_BLINK];
;  4517          4633  4              FORWLRUM[LRUM$L_BLINK] = .BACKLRUM;
;  4518          4634  4              BACKLRUM[LRUM$L_FLINK] = .FORWLRUM;
;  4519          4635  4              DBG$REL_MEMORY(.LRUMPTR);
;  4520          4636  4              EXITLOOP;
;  4521          4637  3              END;
;  4522          4638  3
```

```
;  4523          4639  3          LRUMPTR = .LRUMPTR[LRUM$L_FLINK];
;  4524          4640  2          END;
;  4525          4641  2
;  4526          4642  2
;  4527          4643  2      ! We are all done with the module--now return.
;  4528          4644  2      !
;  4529          4645  2      RETURN;
;  4530          4646  2
;  4531          4647  1      END;


                                    01FC 00000          .ENTRY  DBG$RST_REMOVE, Save R2,R3,R4,R5,R6,R7,R8    ; 4468
                          58 00000000G   00   9E 00002          MOVAB   DBG$REL_MEMORY, R8                   ;
                          57 00000000'   EF   9E 00009          MOVAB   RST$REF_LIST, R7                     ;
                          5E             04   C2 00010          SUBL2   #4, SP                               ;
                          53       04    AC   D0 00013          MOVL    MODRSTPTR, R3                        ; 4525
                      28  A3             02   8A 00017          BICB2   #2, 40(R3)                           ;
                                        7E   D4 0001B          CLRL    LOCKNODE                             ; 4526
                          50            67   D0 0001D          MOVL    RST$REF_LIST, R0                     ; 4527
                          54       04   A0   D0 00020          MOVL    4(R0), R4                            ;
                                        52   D4 00024          CLRL    I                                    ;
                                        13   11 00026          BRB     2$                                   ;
                          50       00 B742   DE 00028 1$:      MOVAL   @RST$REF_LIST[I], R0                 ; 4529
                      04  AE       04   A0   D0 0002D          MOVL    4(R0), LOCKNODE+4                    ;
                                        5E   DD 00032          PUSHL   SP                                   ; 4530
                 00000000G  00          01   FB 00034          CALLS   #1, DBG$STA_LOCK_SYMID               ;
                E9          52          54   F3 0003B 2$:      AOBLEQ  R4, I, 1$                            ; 4527
                            52       08   A3   D0 0003F          MOVL    8(R3), RSTPTR                        ; 4539
                                     08   A3   D4 00043          CLRL    8(R3)                                ; 4540
                                          52   D5 00046 3$:      TSTL    RSTPTR                               ; 4541
                                          33   13 00048          BEQL    7$                                   ;
                            54            62   D0 0004A          MOVL    (RSTPTR), FORWPTR                    ; 4548
                            55       04   A2   7D 0004D          MOVQ    4(RSTPTR), BACKPTR                   ; 4549
                      04    A4            55   D0 00051          MOVL    BACKPTR, 4(FORWPTR)                  ; 4550
                            65            54   D0 00055          MOVL    FORWPTR, (BACKPTR)                   ; 4551
                            02       14   A2   91 00058          CMPB    20(RSTPTR), #2                       ; 4559
                                          03   12 0005C          BNEQ    4$                                   ;
                                     20   A2   D4 0005E          CLRL    32(RSTPTR)                           ; 4561
                                     16   A2   B5 00061 4$:      TSTW    22(RSTPTR)                           ; 4567
                                          07   12 00064          BNEQ    5$                                   ;
                                          52   DD 00066          PUSHL   RSTPTR                               ; 4569
                            68            01   FB 00068          CALLS   #1, DBG$REL_MEMORY                   ;
                                          0B   11 0006B          BRB     6$                                   ;
                            62       0C   A7   D0 0006D 5$:      MOVL    RST$TEMP_LIST, (RSTPTR)              ; 4573
                      0C    A7            52   D0 00071          MOVL    RSTPTR, RST$TEMP_LIST                ; 4575
                                     04   A2   7C 00075          CLRQ    4(RSTPTR)                            ; 4574
                            52            56   D0 00078 6$:      MOVL    NEXTRST, RSTPTR                      ; 4579
                                          C9   11 0007B          BRB     3$                                   ; 4541
                            52       18   A3   D0 0007D 7$:      MOVL    24(R3), SATPTR                       ; 4585
                                     18   A3   D4 00081          CLRL    24(R3)                               ; 4586
                                          52   D5 00084 8$:      TSTL    SATPTR                               ; 4587
                                          0D   13 00086          BEQL    9$                                   ;
                            54            62   D0 00088          MOVL    (SATPTR), NEXTSAT                    ; 4589
                                          52   DD 0008B          PUSHL   SATPTR                               ; 4590
```

```
                68              01 FB 0008D             CALLS   #1, DBG$REL_MEMORY                      ;
                52              54 D0 00090             MOVL    NEXTSAT, SATPTR                         ; 4591
                                EF 11 00093             BRB     8$                                      ; 4587
                        2C      A3 D5 00095 9$:         TSTL    44(R3)                                  ; 4598
                                06 13 00098             BEQL    10$
                        2C      A3 DD 0009A             PUSHL   44(R3)                                  ; 4600
                68              01 FB 0009D             CALLS   #1, DBG$REL_MEMORY
                        2C      A3 D4 000A0 10$:        CLRL    44(R3)                                  ; 4602
                        24      A3 D5 000A3             TSTL    36(R3)                                  ; 4603
                                06 13 000A6             BEQL    11$
                        24      A3 DD 000A8             PUSHL   36(R3)                                  ; 4605
                68              01 FB 000AB             CALLS   #1, DBG$REL_MEMORY
                        24      A3 D4 000AE 11$:        CLRL    36(R3)                                  ; 4607
                50              67 D0 000B1             MOVL    RST$REF_LIST, R0                        ; 4614
                54      04      A0 D0 000B4             MOVL    4(R0), R4
                                52 D4 000B8             CLRL    I
                                13 11 000BA             BRB     13$
                50      00 B742 DE 000BC 12$:           MOVAL   @RST$REF_LIST[I], R0                    ; 4616
        04      AE      04      A0 D0 000C1             MOVL    4(R0), LOCKNODE+4
                                5E DD 000C6             PUSHL   SP                                      ; 4617
00000000G       00              01 FB 000C8             CALLS   #1, DBG$STA_UNLOCK_SYMID
    E9          52              54 F3 000CF 13$:        AOBLEQ  R4, I, 12$                              ; 4614
        FC      A7              53 D1 000D3             CMPL    R3, LRUM$MOST_RECENT                    ; 4624
                                03 12 000D7             BNEQ    14$
                        FC      A7 D4 000D9             CLRL    LRUM$MOST_RECENT
                52      F8      B7 D0 000DC 14$:        MOVL    @LRUM$LISTHEAD, LRUMPTR                 ; 4625
        F8      A7              52 D1 000E0 15$:        CMPL    LRUMPTR, LRUM$LISTHEAD                  ; 4626
                                1B 13 000E4             BEQL    17$
                53      08      A2 D1 000E6             CMPL    8(LRUMPTR), R3                          ; 4628
                                10 12 000EA             BNEQ    16$
                54              62 7D 000EC             MOVQ    (LRUMPTR), FORWLRUM                     ; 4631
        04      A4              55 D0 000EF             MOVL    BACKLRUM, 4(FORWLRUM)                   ; 4633
                65              54 D0 000F3             MOVL    FORWLRUM, (BACKLRUM)                    ; 4634
                                52 DD 000F6             PUSHL   LRUMPTR                                 ; 4635
                68              01 FB 000F8             CALLS   #1, DBG$REL_MEMORY
                                04 000FB             RET                                             ; 4630
                52              62 D0 000FC 16$:        MOVL    (LRUMPTR), LRUMPTR                      ; 4639
                                DF 11 000FF             BRB     15$                                     ; 4626
                                04 00101 17$:           RET                                             ; 4647

; Routine Size:  258 bytes,    Routine Base:  DBG$CODE + 2468
```

```
;  4533          4648  1 GLOBAL ROUTINE DBG$RST_SETMOD(MODNAMEPTR, LENGTH, ALLOCATE_FLAG) =
;  4534          4649  1 !
;  4535          4650  1 ! FUNCTION
;  4536          4651  1 !       This routine marks a specified module as being SET.  The routine accepts
;  4537          4652  1 !       the desired module name and its length as input, looks that name up in
;  4538          4653  1 !       the RST Module Chain, and marks the module as being SET.  Alternatively,
;  4539          4654  1 !       if no module name is specified, all modules in the RST Module Chain are
;  4540          4655  1 !       marked as SET (Shareable image module must be set individually).
;  4541          4656  1 !
;  4542          4657  1 ! INPUTS
;  4543          4658  1 !       MODNAMEPTR - A pointer to the first character of an ASCII string
;  4544          4659  1 !                    containing the module name.  If this pointer is zero, it is
;  4545          4660  1 !                    assumed that all modules should be marked as being SET.
;  4546          4661  1 !
;  4547          4662  1 !       LENGTH  - The length in characters of the module name.
;  4548          4663  1 !
;  4549          4664  1 !       ALLOCATE_FLAG - TRUE if SET MODULE/ALLOCATE was specified.
;  4550          4665  1 !
;  4551          4666  1 ! OUTPUTS
;  4552          4667  1 !       This routine returns the value STS$K_SUCCESS if the module was success-
;  4553          4668  1 !                    fully SET; it returns STS$K_SEVERE if the specified module
;  4554          4669  1 !                    name does not exist in the RST Module Chain.  If all modules
;  4555          4670  1 !                    are SET, STS$K_SUCCESS is always returned.
;  4556          4671  1 !
;  4557          4672  1
;  4558          4673  2     BEGIN
;  4559          4674  2
;  4560          4675  2     LOCAL
;  4561          4676  2         RSTPTR: REF RST$ENTRY;          ! Pointer to the Module RST Entry
;  4562          4677  2
;  4563          4678  2
;  4564          4679  2
;  4565          4680  2     ! Set the flag saying whether it is OK to allocate more memory in order
;  4566          4681  2     ! to set this module. This is true if /ALLOCATE was specifically
;  4567          4682  2     ! requested.
;  4568          4683  2     !
;  4569          4684  2     DBG$GV_CONTROL[DBG$V_CONTROL_ALLOCATE] = .ALLOCATE_FLAG;
;  4570          4685  2
;  4571          4686  2
;  4572          4687  2     ! If a specific module name was specified, we look up the corresponding
;  4573          4688  2     ! Module RST Entry, mark the module as being SET, and build the RST symbol
;  4574          4689  2     ! chain for the module.
;  4575          4690  2     !
;  4576          4691  2     IF .MODNAMEPTR NEQ 0
;  4577          4692  2     THEN
;  4578          4693  3         BEGIN
;  4579          4694  3         RSTPTR = GET_MOD_RST(.MODNAMEPTR, .LENGTH);
;  4580          4695  3         IF .RSTPTR EQL 0 THEN RETURN STS$K_SEVERE;
;  4581          4696  3         RSTPTR[RST$V_MODSET] = TRUE;
;  4582          4697  3         DBG$RST_BUILD(.RSTPTR);
;  4583          4698  3         RETURN STS$K_SUCCESS;
;  4584          4699  2         END;
;  4585          4700  2
;  4586          4701  2
;  4587          4702  2     ! No module name was specified.  Go through the whole RST Module Chain (but
;  4588          4703  2     ! ignore the anonymous module) and mark each module as being SET.  Also
;  4589          4704  2     ! build each such module's RST symbol chain.  Note that we abort this loop
```

```
;  4590          4705  2     ! if the user entered Control-Y DEBUG to stop the SET MODULE command.
;  4591          4706  2     !
;  4592          4707  2     RSTPTR = .RST$START_ADDR;
;  4593          4708  2     WHILE .RSTPTR NEQ 0 DO
;  4594          4709  3         BEGIN
;  4595          4710  3         $ABORT_ON_CONTROL_Y;
;  4596          4711  3         IF NOT .RSTPTR[RST$V_ANONMOD]
;  4597          4712  3         THEN
;  4598          4713  4             BEGIN
;  4599          4714  4             IF NOT .RSTPTR[RST$V_SHARE_IMAGE]
;  4600          4715  4             THEN
;  4601          4716  5                 BEGIN
;  4602          4717  5                 RSTPTR[RST$V_MODSET] = TRUE;
;  4603          4718  5                 DBG$RST_BUILD(.RSTPTR);
;  4604          4719  4                 END;
;  4605          4720  3             END;
;  4606          4721  3
;  4607          4722  3         RSTPTR = .RSTPTR[RST$L_NXTMODPTR];
;  4608          4723  2         END;
;  4609          4724  2
;  4610          4725  2     RETURN STS$K_SUCCESS;
;  4611          4726  2
;  4612          4727  1     END;


                                                0004 00000          .ENTRY  DBG$RST_SETMOD, Save R2                         ; 4648
00000000G  00         01          07      0C    AC   F0 00002          INSV    ALLOCATE_FLAG, #7, #1, DBG$GV_CONTROL           ; 4684
                                          04    AC   D5 0000C          TSTL    MODNAMEPTR                                      ; 4691
                                          1F    13 0000F               BEQL    2$
                            7E            04    AC   7D 00011          MOVQ    MODNAMEPTR, -(SP)                               ; 4694
                      0000V CF                    02   FB 00015          CALLS   #2, GET_MOD_RST
                            52                    50   D0 0001A          MOVL    R0, RSTPTR
                                                  04   12 0001D          BNEQ    1$                                      ; 4695
                            50                    04   D0 0001F          MOVL    #4, R0
                                                       04 00022          RET
                        28  A2                    01   88 00023 1$:     BISB2   #1, 40(RSTPTR)                          ; 4696
                                                  52   DD 00027          PUSHL   RSTPTR                                  ; 4697
                      E226  CF                    01   FB 00029          CALLS   #1, DBG$RST_BUILD
                                                  39   11 0002E          BRB     6$                                      ; 4698
                            52 00000000'          EF   D0 00030 2$:     MOVL    RST$START_ADDR, RSTPTR                  ; 4707
                                                  30   13 00037 3$:     BEQL    6$                                      ; 4708
                   0D 00000000G   00              01   E1 00039          BBC     #1, DBG$GV_CONTROL+1, 4$                ; 4709
                                  000280E8        8F   DD 00041          PUSHL   #164072
                      00000000G   00              01   FB 00047          CALLS   #1, LIB$SIGNAL
                   10        28   A2              02   E0 0004E 4$:     BBS     #2, 40(RSTPTR), 5$                      ; 4711
                   08        28   A2              04   E0 00053          BBS     #4, 40(RSTPTR), 5$                      ; 4714
                             28   A2              01   88 00058          BISB2   #1, 40(RSTPTR)                          ; 4717
                                                  52   DD 0005C          PUSHL   RSTPTR                                  ; 4718
                            E1F1  CF              01   FB 0005E          CALLS   #1, DBG$RST_BUILD
                                                  52   10   A2 00063 5$:     MOVL    16(RSTPTR), RSTPTR                      ; 4722
                                                  CE   11 00067          BRB     3$                                      ; 4708
                                                  50   01   D0 00069 6$:     MOVL    #1, R0                                  ; 4725
                                                       04   0006C          RET                                             ; 4727
```

; Routine Size: 109 bytes,   Routine Base: DBG$CODE + 256A

```
;4614     4728  1 GLOBAL ROUTINE DBG$RST_SETSCOPE(PNVEC, ERRINDEX, SET_SCOPE_MODULE): NOVALUE =
;4615     4729  1 !
;4616     4730  1 ! FUNCTION
;4617     4731  1 !       This routine makes a new set of scopes available to the symbol table
;4618     4732  1 !       access routines (DBG$STA_GETSYMBOL in particular).  It is thus called
;4619     4733  1 !       in the course of processing the SET SCOPE command.  It accepts as in-
;4620     4734  1 !       put a vector of pointers to Pathname Descriptors for the scopes speci-
;4621     4735  1 !       fied on the SET SCOPE command.  It calls DBG$RST_BLDSCOPE_LIST to
;4622     4736  1 !       build a linked list of the corresponding Scope Entries, and makes
;4623     4737  1 !       SCOPE$LIST point to this list.
;4624     4738  1 !
;4625     4739  1 ! INPUTS
;4626     4740  1 !       PNVEC   - A vector of pointers to Pathname Descriptors for the scopes
;4627     4741  1 !                 specified on the SET SCOPE command.  PNVEC[0] gives the number
;4628     4742  1 !                 of such pointers and PNVEC[1] through PNVEC[.PNVEC[0]] contain
;4629     4743  1 !                 the pointers to the Pathname Descriptors.  The CANCEL SCOPE
;4630     4744  1 !                 command is handled by making PNVEC[0] zero.
;4631     4745  1 !
;4632     4746  1 !       ERRINDEX - The address of a longword location to receive an "error
;4633     4747  1 !                  index"; see description below.
;4634     4748  1 !
;4635     4749  1 !       SET_SCOPE_MODULE - An optional 3rd parameter which if present and
;4636     4750  1 !                          TRUE means we were called from SET SCOPE/MODULE.
;4637     4751  1 !
;4638     4752  1 ! OUTPUTS
;4639     4753  1 !       ERRINDEX - If some pathname is found to be invalid, the PNVEC index of
;4640     4754  1 !                  that pathname is returned to ERRINDEX.  The calling routine
;4641     4755  1 !                  can then use ERRINDEX to produce an error message.  If all
;4642     4756  1 !                  scope pathnames are correct, zero is returned to ERRINDEX.
;4643     4757  1 !
;4644     4758  1 !       No value is returned.
;4645     4759  1 !
;4646     4760  1
;4647     4761  2     BEGIN
;4648     4762  2
;4649     4763  2     MAP
;4650     4764  2         PNVEC: REF VECTOR[,LONG],       ! Pointer to vector of Pathname Descrip-
;4651     4765  2                                         !     tor pointers
;4652     4766  2         ERRINDEX: REF VECTOR[1];        ! Pointer to error index longword
;4653     4767  2
;4654     4768  2     LOCAL
;4655     4769  2         LOCKNODE: DBG$LINK_NODE,        ! Node used to lock or unlock SYMIDs
;4656     4770  2         MODULE_FLAG,                    ! TRUE if called from SET SCOPE/MODULE
;4657     4771  2         NEW_SCOPE_LIST,                 ! Pointer to new scope list being built
;4658     4772  2         PREVSCOPE: REF SCOPE$ENTRY,     ! Pointer to the scope entry before the
;4659     4773  2                                         !     present scope entry
;4660     4774  2         SCOPE: REF SCOPE$ENTRY;         ! Pointer to the present scope entry
;4661     4775  2
;4662     4776  2     BUILTIN
;4663     4777  2         ACTUALCOUNT;
;4664     4778  2
;4665     4779  2
;4666     4780  2     ! Initialize the flag that says whether we were called from
;4667     4781  2     ! SET SCOPE/MODULE. This is just passed along to the BLDSCOPE routine
;4668     4782  2     ! that actually processes the SET SCOPE command.
;4669     4783  2     !
;4670     4784  2     IF ACTUALCOUNT() LSS 3
```

```
;  4671          4785  2      THEN
;  4672          4786  2          MODULE_FLAG = FALSE
;  4673          4787  2      ELSE
;  4674          4788  2          MODULE_FLAG = .SET_SCOPE_MODULE;
;  4675          4789  2
;  4676          4790  2
;  4677          4791  2      NEW_SCOPE_LIST = DBG$RST_BLDSCOPE_LIST(.PNVEC, ERRINDEX[0],
;  4678          4792  2                                             TRUE, .MODULE_FLAG);
;  4679          4793  2      IF .ERRINDEX[0] NEQ 0 THEN RETURN;
;  4680          4794  2
;  4681          4795  2
;  4682          4796  2      ! Lock the scope's RST entry in memory.
;  4683          4797  2      !
;  4684          4798  2      !
;  4685          4799  2      SCOPE = .NEW_SCOPE_LIST;
;  4686          4800  2      WHILE .SCOPE NEQ 0 DO
;  4687          4801  3          BEGIN
;  4688          4802  3          IF .SCOPE[SCOPE$L_STATE] EQL SCOPE$K_NORMAL
;  4689          4803  3          THEN
;  4690          4804  4              BEGIN
;  4691          4805  4              LOCKNODE[DBG$L_LINK_NODE_LINK] = 0;
;  4692          4806  4              LOCKNODE[DBG$L_LINK_NODE_VALUE] = .SCOPE[SCOPE$L_RSTPTR];
;  4693          4807  4              DBG$STA_LOCK_SYMID(LOCKNODE);
;  4694          4808  3              END;
;  4695          4809  3
;  4696          4810  3          PREVSCOPE = .SCOPE;
;  4697          4811  3          SCOPE = .PREVSCOPE[SCOPE$L_FLINK];
;  4698          4812  2          END;
;  4699          4813  2
;  4700          4814  2
;  4701          4815  2      ! We have now built the new scope list successfully.
;  4702          4816  2      ! Append two more scope entries to the end of the new scope list: one for
;  4703          4817  2      ! all SET modules and one for the global (GST) scope.
;  4704          4818  2      !
;  4705          4819  2      SCOPE = DBG$GET_MEMORY(SCOPE$K_ENTSIZE);
;  4706          4820  2      SCOPE[SCOPE$L_FLINK] = 0;
;  4707          4821  2      SCOPE[SCOPE$L_STATE] = SCOPE$K_SETMODS;
;  4708          4822  2      SCOPE[SCOPE$L_RSTPTR] = 0;
;  4709          4823  2      SCOPE[SCOPE$L_MODPTR] = 0;
;  4710          4824  2      PREVSCOPE[SCOPE$L_FLINK] = .SCOPE;
;  4711          4825  2      PREVSCOPE = .SCOPE;
;  4712          4826  2      SCOPE = DBG$GET_MEMORY(SCOPE$K_ENTSIZE);
;  4713          4827  2      SCOPE[SCOPE$L_FLINK] = 0;
;  4714          4828  2      SCOPE[SCOPE$L_STATE] = SCOPE$K_GLOBAL;
;  4715          4829  2      SCOPE[SCOPE$L_RSTPTR] = 0;
;  4716          4830  2      SCOPE[SCOPE$L_MODPTR] = 0;
;  4717          4831  2      PREVSCOPE[SCOPE$L_FLINK] = .SCOPE;
;  4718          4832  2
;  4719          4833  2
;  4720          4834  2      ! Then loop through the old scope list and release all the old scope entries
;  4721          4835  2      ! to the free memory pool.  Note that we unlock normal scope RST entries.
;  4722          4836  2      !
;  4723          4837  2      WHILE .SCOPE$LIST NEQ 0 DO
;  4724          4838  3          BEGIN
;  4725          4839  3          IF .SCOPE$LIST[SCOPE$L_STATE] EQL SCOPE$K_NORMAL
;  4726          4840  3          THEN
;  4727          4841  4              BEGIN
```

```
;  4728     4842  4              LOCKNODE[DBG$L_LINK_NODE_LINK] = 0;
;  4729     4843  4              LOCKNODE[DBG$L_LINK_NODE_VALUE] = .SCOPE$LIST[SCOPE$L_RSTPTR];
;  4730     4844  4              DBG$STA_UNLOCK_SYMID(LOCKNODE);
;  4731     4845  3              END;
;  4732     4846  3
;  4733     4847  3          PREVSCOPE = .SCOPE$LIST;
;  4734     4848  3          SCOPE$LIST = .PREVSCOPE[SCOPE$L_FLINK];
;  4735     4849  3          DBG$REL_MEMORY(.PREVSCOPE);
;  4736     4850  2          END;
;  4737     4851  2
;  4738     4852  2
;  4739     4853  2      ! Set the start address of the new scope list in SCOPE_LIST; that makes it
;  4740     4854  2      ! official.  Then return.
;  4741     4855  2      !
;  4742     4856  2      SCOPE$LIST = .NEW_SCOPE_LIST;
;  4743     4857  2      RETURN;
;  4744     4858  2
;  4745     4859  1      END;


                                007C 00000          .ENTRY  DBG$RST_SETSCOPE, Save R2,R3,R4,R5,R6       ; 4728
56 00000000G   00    9E 00002          MOVAB   DBG$GET_MEMORY, R6                                          ;
55 00000000'   EF    9E 00009          MOVAB   SCOPE$LIST, R5                                              ;
5E             08    C2 00010          SUBL2   #8, SP                                                      ;
03             6C    91 00013          CMPB    (AP), #3                                                    ; 4784
               04    1E 00016          BGEQU   1$                                                          ;
               50    D4 00018          CLRL    MODULE_FLAG                                                 ; 4786
               04    11 0001A          BRB     2$                                                          ;
50       0C    AC    D0 0001C 1$:      MOVL    SET_SCOPE_MODULE, MODULE_FLAG                               ; 4788
               50    DD 00020 2$:      PUSHL   MODULE_FLAG                                                 ; 4792
               01    DD 00022          PUSHL   #1                                                          ; 4791
7E             04    AC    7D 00024          MOVQ    PNVEC, -(SP)                                        ;
F1CD           CF          04    FB 00028          CALLS   #4, DBG$RST_BLDSCOPE_LIST                     ;
               54          50    D0 0002D          MOVL    R0, NEW_SCOPE_LIST                            ;
                     08    BC    D5 00030          TSTL    @ERRINDEX                                     ; 4793
               7D          12 00033          BNEQ    9$                                                  ;
               52          54    D0 00035          MOVL    NEW_SCOPE_LIST, SCOPE                         ; 4799
               1E          13 00038 3$:      BEQL    5$                                                  ; 4800
               01    04    A2    D1 0003A          CMPL    4(SCOPE), #1                                  ; 4802
               10          12 0003E          BNEQ    4$                                                  ;
               6E          D4 00040          CLRL    LOCKNODE                                            ; 4805
04             AE    08    A2    D0 00042          MOVL    8(SCOPE), LOCKNODE+4                          ; 4806
               5E          DD 00047          PUSHL   SP                                                  ; 4807
00000000G      00          01    FB 00049          CALLS   #1, DBG$STA_LOCK_SYMID                        ;
               53          52    D0 00050 4$:      MOVL    SCOPE, PREVSCOPE                              ; 4810
               52          63    D0 00053          MOVL    (PREVSCOPE), SCOPE                            ; 4811
               E0          11 00056          BRB     3$                                                  ; 4800
               04    DD 00058 5$:      PUSHL   #4                                                          ; 4819
               66          01    FB 0005A          CALLS   #1, DBG$GET_MEMORY                            ;
               52          50    D0 0005D          MOVL    R0, SCOPE                                     ;
               62          D4 00060          CLRL    (SCOPE)                                             ; 4820
04             A2          04    7D 00062          MOVQ    #4, 4(SCOPE)                                  ; 4821
                     0C    A2    D4 00066          CLRL    12(SCOPE)                                     ; 4823
               63          52    D0 00069          MOVL    SCOPE, (PREVSCOPE)                            ; 4824
```

```
                53              52  D0 0006C          MOVL    SCOPE, PREVSCOPE                        ; 4825
                                04  DD 0006F          PUSHL   #4                                      ; 4826
                66              01  FB 00071          CALLS   #1, DBG$GET_MEMORY                      ;
                52              50  D0 00074          MOVL    R0, SCOPE                               ;
                                62  D4 00077          CLRL    (SCOPE)                                 ; 4827
             04 A2              03  7D 00079          MOVQ    #3, 4(SCOPE)                            ; 4828
                         0C     A2  D4 0007D          CLRL    12(SCOPE)                               ; 4830
                63              52  D0 00080          MOVL    SCOPE, (PREVSCOPE)                      ; 4831
                50              65  D0 00083 6$:      MOVL    SCOPE$LIST, R0                          ; 4837
                                27  13 00086          BEQL    8$                                      ;
                01          04  A0  D1 00088          CMPL    4(R0), #1                               ; 4839
                                10  12 0008C          BNEQ    7$                                      ;
                                6E  D4 0008E          CLRL    LOCKNODE                                ; 4842
             04 AE          08  A0  D0 00090          MOVL    8(R0), LOCKNODE+4                       ; 4843
                                5E  DD 00095          PUSHL   SP                                      ; 4844
00000000G       00              01  FB 00097          CALLS   #1, DBG$STA_UNLOCK_SYMID                ;
                53              65  D0 0009E 7$:      MOVL    SCOPE$LIST, PREVSCOPE                   ; 4847
                65              63  D0 000A1          MOVL    (PREVSCOPE), SCOPE$LIST                 ; 4848
                                53  DD 000A4          PUSHL   PREVSCOPE                               ; 4849
00000000G       00              01  FB 000A6          CALLS   #1, DBG$REL_MEMORY                      ;
                                D4  11 000AD          BRB     6$                                      ; 4837
                65              54  D0 000AF 8$:      MOVL    NEW_SCOPE_LIST, SCOPE$LIST              ; 4854
                                    04 000B2 9$:      RET                                             ; 4859

; Routine Size:  179 bytes,    Routine Base:  DBG$CODE + 25D7
```

```
;4747     4860  1 ROUTINE BLDSCOPE_ERROR_HANDLER(SIGARG, MECHARG, ENBLARG) =
;4748     4861  1 !
;4749     4862  1 ! FUNCTION
;4750     4863  1 !       This routine is the error handler for the DBG$RST_BLDSCOPE_LIST routine.
;4751     4864  1 !       All it does is to handle the DBG$_BADSCOPE and DBG$_NOFREE errors by
;4752     4865  1 !       releasing all Scope Entries accumulated on NEW_SCOPE_LIST back to the
;4753     4866  1 !       memory pool so they are not lost.  For DBG$_BADSCOPE, the handler then
;4754     4867  1 !       unwinds so that control is returned to the routine that called routine
;4755     4868  1 !       DBG$RST_BLDSCOPE_LIST.  Otherwise, it just resignals the error.
;4756     4869  1 !
;4757     4870  1 ! INPUTS
;4758     4871  1 !       SIGARG  - The signal argument vector.
;4759     4872  1 !
;4760     4873  1 !       MECHARG - The mechanism argument vector.
;4761     4874  1 !
;4762     4875  1 !       ENBLARG - The enable argument vector.  ENBLARG[1] is assumed to
;4763     4876  1 !                 contains the address of NEW_SCOPE_LIST.
;4764     4877  1 !
;4765     4878  1 ! OUTPUTS
;4766     4879  1 !       For the DBG$_BADSCOPE error, this handler unwinds the call stack.
;4767     4880  1 !                 For all other errors, it resignals.
;4768     4881  1 !
;4769     4882  1
;4770     4883  2     BEGIN
;4771     4884  2
;4772     4885  2     MAP
;4773     4886  2         SIGARG: REF VECTOR[,LONG],      ! Signal argument vector
;4774     4887  2         ENBLARG: REF VECTOR[,LONG];     ! Enable declaration argument vector
;4775     4888  2
;4776     4889  2     BIND
;4777     4890  2         NEW_SCOPE_LIST = .ENBLARG[1];   ! Address of pointer to new scope list
;4778     4891  2
;4779     4892  2     LOCAL
;4780     4893  2         RELSCOPE,                       ! Pointer to scope entry being released
;4781     4894  2         SCOPE: REF SCOPE$ENTRY;         ! Pointer to the next scope list entry
;4782     4895  2
;4783     4896  2
;4784     4897  2
;4785     4898  2     ! If the condition code is anything other than DBG$_BADSCOPE or DBG$_NOFREE,
;4786     4899  2     ! just resignal the error.
;4787     4900  2     !
;4788     4901  2     IF .SIGARG[1] NEQ DBG$_BADSCOPE AND
;4789     4902  2        .SIGARG[1] NEQ DBG$_NOUNIQUE AND
;4790     4903  2        .SIGARG[1] NEQ DBG$_NOFREE
;4791     4904  2     THEN
;4792     4905  2         RETURN SS$_RESIGNAL;
;4793     4906  2
;4794     4907  2
;4795     4908  2     ! It is one of the errors we handle.  Loop through the new scope list and
;4796     4909  2     ! release each Scope Entry on it back to the free memory pool.
;4797     4910  2     !
;4798     4911  2     SCOPE = .NEW_SCOPE_LIST;
;4799     4912  2     WHILE .SCOPE NEQ 0 DO
;4800     4913  3         BEGIN
;4801     4914  3         RELSCOPE = .SCOPE;
;4802     4915  3         SCOPE = .SCOPE[SCOPE$L_FLINK];
;4803     4916  3         DBG$REL_MEMORY(.RELSCOPE);
```

```
;  4804          4917  2          END;
;  4805          4918  2
;  4806          4919  2
;  4807          4920  2      ! For DBG$_BADSCOPE, unwind the stack to return control to the caller of
;  4808          4921  2      ! DBG$RST_SETSCOPE.  For DBG$_NOFREE, we simply resignal the error.
;  4809          4922  2      !
;  4810          4923  2      IF .SIGARG[1] EQL DBG$_BADSCOPE THEN SETUNWIND();
;  4811          4924  2      RETURN SS$_RESIGNAL;
;  4812          4925  2
;  4813          4926  1      END;


                                001C 00000 BLDSCOPE_ERROR_HANDLER:
                                           .WORD   Save R2,R3,R4                    ; 4880
                50          0C  AC  D0 00002       MOVL    ENBLARG, R0             ; 4890
                52          04  AC  D0 00006       MOVL    SIGARG, R2              ; 4901
000281D8        8F          04  A2  D1 0000A       CMPL    4(R2), #164312
                                14  13 00012       BEQL    1$
000281F0        8F          04  A2  D1 00014       CMPL    4(R2), #164336          ; 4902
                                0A  13 0001C       BEQL    1$
00028332        8F          04  A2  D1 0001E       CMPL    4(R2), #164658          ; 4903
                                2C  12 00026       BNEQ    4$
                53          04  B0  D0 00028 1$:    MOVL    @4(R0), SCOPE           ; 4911
                                53  D5 0002C 2$:    TSTL    SCOPE                   ; 4912
                                11  13 0002E       BEQL    3$
                54              53  D0 00030       MOVL    SCOPE, RELSCOPE         ; 4914
                53              63  D0 00033       MOVL    (SCOPE), SCOPE          ; 4915
                                54  DD 00036       PUSHL   RELSCOPE                ; 4916
00000000G       00              01  FB 00038       CALLS   #1, DBG$REL_MEMORY
                                EB  11 0003F       BRB     2$                      ; 4912
000281D8        8F          04  A2  D1 00041 3$:    CMPL    4(R2), #164312          ; 4923
                                09  12 00049       BNEQ    4$
                                7E  7C 0004B       CLRQ    -(SP)
00000000G       00              02  FB 0004D       CALLS   #2, SYS$UNWIND
                50      0918    8F  3C 00054 4$:    MOVZWL  #2328, R0               ; 4924
                                04  00059       RET                             ; 4926

; Routine Size:  90 bytes,    Routine Base:  DBG$CODE + 268A
```

```
;  4815          4927  1 ROUTINE BUILD_LABEL_RST(DSTPTR) =
;  4816          4928  1 !
;  4817          4929  1 ! FUNCTION
;  4818          4930  1 !       This routine builds a skeleton Label RST Entry.  The main thing it does,
;  4819          4931  1 !       however, is to determine if a label is a numeric label.  If the label is
;  4820          4932  1 !       numeric, a memory block is allocated to hold both the Label RST Entry
;  4821          4933  1 !       and a dummy DST record for that label.  The dummy DST record is set up
;  4822          4934  1 !       so that the name field contains "%LABEL nnn" where nnn is the label num-
;  4823          4935  1 !       ber stripped of all leading zeroes.  This allows the Debugger to search
;  4824          4936  1 !       for "%LABEL nnn" as it searches for any other symbol without making any
;  4825          4937  1 !       special provisions for the "%LABEL" part (other than when parsing).
;  4826          4938  1 !
;  4827          4939  1 ! INPUTS
;  4828          4940  1 !       DSTPTR  - A pointer to the original label DST record for the label.
;  4829          4941  1 !
;  4830          4942  1 ! OUTPUTS
;  4831          4943  1 !       A pointer to a skeleton Label RST Entry is returned as the routine's
;  4832          4944  1 !                 value.  The RST$L_DSTPTR field is filled in correctly in this
;  4833          4945  1 !                 entry, but all other fields are empty.  If the label is numer-
;  4834          4946  1 !                 ic, this pointer points to a dummy DST record with a modified
;  4835          4947  1 !                 label name; otherwise it points to the original DST record.
;  4836          4948  1 !
;  4837          4949  1
;  4838          4950  2     BEGIN
;  4839          4951  2
;  4840          4952  2     MAP
;  4841          4953  2         DSTPTR: REF DST$RECORD;          ! Pointer to original label DST record
;  4842          4954  2
;  4843          4955  2     LOCAL
;  4844          4956  2         FIRSTLOC,                        ! Location of first non-zero digit
;  4845          4957  2         J,                               ! Index into new label name string
;  4846          4958  2         NAMEPTR: REF VECTOR[,BYTE],      ! Pointer to original name string
;  4847          4959  2         NEWDST: REF DST$RECORD,          ! Pointer to new "dummy" DST record
;  4848          4960  2         NEWNAME: REF VECTOR[,BYTE],      ! Pointer to new label name string
;  4849          4961  2         NUMLBL_FLAG,                     ! Set to TRUE if label is numeric
;  4850          4962  2         RSTPTR: REF RST$ENTRY;           ! Pointer to new Label RST Entry
;  4851          4963  2
;  4852          4964  2
;  4853          4965  2
;  4854          4966  2     ! Determine if the input label's name is fully numeric.  Also determine the
;  4855          4967  2     ! location of the first non-zero digit in the name.
;  4856          4968  2     !
;  4857          4969  2     NAMEPTR = DSTPTR[DST$B_NAME];
;  4858          4970  2     FIRSTLOC = 0;
;  4859          4971  2     NUMLBL_FLAG = TRUE;
;  4860          4972  2     INCR I FROM 1 TO .NAMEPTR[0] DO
;  4861          4973  3         BEGIN
;  4862          4974  4         IF (.NAMEPTR[.I] LSS '0') OR (.NAMEPTR[.I] GTR '9')
;  4863          4975  3         THEN
;  4864          4976  4             BEGIN
;  4865          4977  4             NUMLBL_FLAG = FALSE;
;  4866          4978  4             EXITLOOP;
;  4867          4979  3             END;
;  4868          4980  3
;  4869          4981  3         IF (.NAMEPTR[.I] NEQ '0') AND (.FIRSTLOC EQL 0) THEN FIRSTLOC = .I;
;  4870          4982  2         END;
;  4871          4983  2
```

```
;  4872          4984  2     IF .FIRSTLOC EQL 0 THEN FIRSTLOC = .NAMEPTR[0];
;  4873          4985  2
;  4874          4986  2
;  4875          4987  2     ! If the label is not numeric, get a memory block for the Label RST Entry
;  4876          4988  2     ! only, put the DST pointer in the RST$L_DSTPTR field, and return.
;  4877          4989  2     !
;  4878          4990  3     IF (NOT .NUMLBL_FLAG) OR (.NAMEPTR[0] EQL 0)
;  4879          4991  2     THEN
;  4880          4992  3         BEGIN
;  4881          4993  3         RSTPTR = DBG$GET_MEMORY(RST$K_LBLENTSIZ);
;  4882          4994  3         RSTPTR[RST$L_DSTPTR] = .DSTPTR;
;  4883          4995  3         RETURN .RSTPTR;
;  4884          4996  2         END;
;  4885          4997  2
;  4886          4998  2
;  4887          4999  2     ! The label is numeric.  Allocate a memory block large enough to hold both
;  4888          5000  2     ! the Label RST Entry and a dummy DST record for the label.  The name in
;  4889          5001  2     ! this dummy DST record is set to "%LABEL nnn" where nnn is all the digits
;  4890          5002  2     ! of the label number except for leading zeroes.  We then return.
;  4891          5003  2     !
;  4892          5004  2     RSTPTR = DBG$GET_MEMORY(RST$K_LBLENTSIZ + (.NAMEPTR[0] - .FIRSTLOC + 19)/4);
;  4893          5005  2     NEWDST = .RSTPTR + 4*RST$K_LBLENTSIZ;
;  4894          5006  2     RSTPTR[RST$L_DSTPTR] = .NEWDST;
;  4895          5007  2     NEWDST[DST$B_LENGTH] = .NAMEPTR[0] - .FIRSTLOC + 15;
;  4896          5008  2     NEWDST[DST$B_TYPE] = DST$K_LABEL;
;  4897          5009  2     NEWDST[DST$B_VFLAGS] = 0;
;  4898          5010  2     NEWDST[DST$L_VALUE] = .DSTPTR[DST$L_VALUE];
;  4899          5011  2     NEWNAME = NEWDST[DST$B_NAME];
;  4900          5012  2     NEWNAME[1] = '%';
;  4901          5013  2     NEWNAME[2] = 'L';
;  4902          5014  2     NEWNAME[3] = 'A';
;  4903          5015  2     NEWNAME[4] = 'B';
;  4904          5016  2     NEWNAME[5] = 'E';
;  4905          5017  2     NEWNAME[6] = 'L';
;  4906          5018  2     NEWNAME[7] = ' ';
;  4907          5019  2     J = 7;
;  4908          5020  2     INCR I FROM .FIRSTLOC TO .NAMEPTR[0] DO
;  4909          5021  3         BEGIN
;  4910          5022  3         J = .J + 1;
;  4911          5023  3         NEWNAME[.J] = .NAMEPTR[.I];
;  4912          5024  2         END;
;  4913          5025  2
;  4914          5026  2     NEWNAME[0] = .J;
;  4915          5027  2     RETURN .RSTPTR;
;  4916          5028  2
;  4917          5029  1     END;


                              00FC 00000 BUILD_LABEL_RST:
                                                 .WORD   Save R2,R3,R4,R5,R6,R7                  ;  4927
                57 00000000G   00    9E 00002    MOVAB   DBG$GET_MEMORY, R7                      ;
                55         04  AC    D0 00009    MOVL    DSTPTR, R5                              ;  4969
                52         07  A5    9E 0000D    MOVAB   7(R5), NAMEPTR                          ;
                               53    D4 00011    CLRL    FIRSTLOC                                ;  4970
```

```
                51            01  D0 00013          MOVL    #1, NUMLBL_FLAG                 ; 4971
                56            62  9A 00016          MOVZBL  (NAMEPTR), R6                   ; 4972
                              50  D4 00019          CLRL    I
                              1D  11 0001B          BRB     4$
                30          6042  91 0001D 1$:      CMPB    (I)[NAMEPTR], #48               ; 4974
                              06  1F 00021          BLSSU   2$
                39          6042  91 00023          CMPB    (I)[NAMEPTR], #57
                              04  1B 00027          BLEQU   3$
                              51  D4 00029 2$:      CLRL    NUMLBL_FLAG                     ; 4977
                              11  11 0002B          BRB     5$                              ; 4976
                30          6042  91 0002D 3$:      CMPB    (I)[NAMEPTR], #48               ; 4981
                              07  13 00031          BEQL    4$
                              53  D5 00033          TSTL    FIRSTLOC
                              03  12 00035          BNEQ    4$
                53              50  D0 00037          MOVL    I, FIRSTLOC
          DF    50              56  F3 0003A 4$:      AOBLEQ  R6, I, 1$                       ; 4972
                              53  D5 0003E 5$:      TSTL    FIRSTLOC                        ; 4984
                              03  12 00040          BNEQ    6$
                53              56  D0 00042          MOVL    R6, FIRSTLOC
                04              51  E9 00045 6$:      BLBC    NUMLBL_FLAG, 7$                 ; 4990
                              56  D5 00048          TSTL    R6
                              0E  12 0004A          BNEQ    8$
                              07  DD 0004C 7$:      PUSHL   #7                              ; 4993
                67              01  FB 0004E          CALLS   #1, DBG$GET_MEMORY
                51              50  D0 00051          MOVL    R0, RSTPTR
          0C    A1              55  D0 00054          MOVL    R5, 12(RSTPTR)                  ; 4994
                              51  11 00058          BRB     11$                             ; 4995
          54                56  53  C3 0005A 8$:      SUBL3   FIRSTLOC, R6, R4                ; 5004
                              50  13  A4  9E 0005E          MOVAB   19(R4), R0
                              50  04  C6  00062          DIVL2   #4, R0
                                  07  A0  9F 00065          PUSHAB  7(R0)
                              67      01  FB 00068          CALLS   #1, DBG$GET_MEMORY
                              51      50  D0 0006B          MOVL    R0, RSTPTR
                              50  1C  A1  9E 0006E          MOVAB   28(R1), NEWDST          ; 5005
                    0C        A1      50  D0 00072          MOVL    NEWDST, 12(RSTPTR)      ; 5006
          80                54      0F  81 00076          ADDB3   #15, R4, (NEWDST)+      ; 5007
                              80  BB  8F  9B 0007A          MOVZBW  #187, (NEWDST)+         ; 5008
                              80  03  A5  D0 0007E          MOVL    3(R5), (NEWDST)+        ; 5010
                    01        A0  42414C25  8F  D0 00082          MOVL    #1111575589, 1(NEWNAME) ; 5012
                    05        A0      4C45  8F  B0 0008A          MOVW    #19525, 5(NEWNAME)      ; 5016
                    07        A0            20  90 00090          MOVB    #32, 7(NEWNAME)         ; 5018
                              54            07  D0 00094          MOVL    #7, J                   ; 5019
                              55      FF    A3  9E 00097          MOVAB   -1(R3), I               ; 5020
                                            07  11 0009B          BRB     10$
                                            54  D6 0009D 9$:      INCL    J                       ; 5022
                        6440                6542  90 0009F          MOVB    (I)[NAMEPTR], (J)[NEWNAME]      ; 5023
          F5            55                  56  F3 000A4 10$:     AOBLEQ  R6, I, 9$               ; 5020
                        60                  54  90 000A8          MOVB    J, (NEWNAME)            ; 5026
                        50                  51  D0 000AB 11$:     MOVL    RSTPTR, R0              ; 5027
                                                04 000AE          RET                             ; 5029

; Routine Size:  175 bytes,    Routine Base:  DBG$CODE + 26E4
```

```
;  4919          5030  1 ROUTINE COPY_COMP_LIST(OLDPTR): NOVALUE =
;  4920          5031  1 !
;  4921          5032  1 ! FUNCTION
;  4922          5033  1 !       This routine copies a record component list from a component list
;  4923          5034  1 !       memory block to memory inside a Data Type RST Entry memory block.
;  4924          5035  1 !       This is done because component lists must be accumulated in separate
;  4925          5036  1 !       memory blocks which can be dynamically grown, but once accumulated,
;  4926          5037  1 !       they are packed into the exact amount of space they need in the Type
;  4927          5038  1 !       RST Entry.  Putting the component list in the same memory block as
;  4928          5039  1 !       the Type RST Entry also simplifies keeping reference counts and the
;  4929          5040  1 !       deallocation of RST entries.
;  4930          5041  1 !
;  4931          5042  1 !       This routine calls COPY_VARIANT_LIST which in turn calls this routine
;  4932          5043  1 !       recursively.  This recursion corresponds to the nesting of records and
;  4933          5044  1 !       variants in the DST.
;  4934          5045  1 !
;  4935          5046  1 ! INPUTS
;  4936          5047  1 !       OLDPTR  - A pointer to the component list memory block to be copied.
;  4937          5048  1 !                 This memory block is released to the free memory pool after
;  4938          5049  1 !                 being copied.
;  4939          5050  1 !
;  4940          5051  1 !       TYPRSTLOC is an implicit input.  It specifies where the new copy should
;  4941          5052  1 !                 be placed in the Type RST Entry memory block.
;  4942          5053  1 !
;  4943          5054  1 ! OUTPUTS
;  4944          5055  1 !       TYPRSTLOC is an implicit output.  It is updated to point to the next
;  4945          5056  1 !                 available location in the Type RST Entry memory block.
;  4946          5057  1 !
;  4947          5058  1 !       No value is returned.
;  4948          5059  1 !
;  4949          5060  1
;  4950          5061  2     BEGIN
;  4951          5062  2
;  4952          5063  2     MAP
;  4953          5064  2         OLDPTR: REF VECTOR[,LONG];      ! Pointer to the list to be copied
;  4954          5065  2
;  4955          5066  2     LOCAL
;  4956          5067  2         NEWPTR: REF VECTOR[,LONG],      ! Pointer to the new copy of the list
;  4957          5068  2         RPTR: REF RST$ENTRY;            ! Pointer to component RST entry
;  4958          5069  2
;  4959          5070  2
;  4960          5071  2
;  4961          5072  2     ! Allocate space for the new copy in the Data Type RST Entry.  Then copy the
;  4962          5073  2     ! old list over to the new location.  Copy any Variant RST Entries into the
;  4963          5074  2     ! Type RST Entry by calling COPY_VARIANT_LIST recursively.  Also change the
;  4964          5075  2     ! kind of all data record components to RST$K_TYPCOMP and set their up-scope
;  4965          5076  2     ! pointers to point to the Type RST Entry.
;  4966          5077  2     !
;  4967          5078  2     NEWPTR = .TYPRSTLOC;
;  4968          5079  2     TYPRSTLOC = TYPRSTLOC[.OLDPTR[0] + 1];
;  4969          5080  2     NEWPTR[0] = .OLDPTR[0];
;  4970          5081  2     INCR I FROM 1 TO .OLDPTR[0] DO
;  4971          5082  3         BEGIN
;  4972          5083  3         RPTR = .OLDPTR[.I];
;  4973          5084  3         IF .RPTR[RST$B_KIND] EQL RST$K_VARIANT
;  4974          5085  3         THEN
;  4975          5086  3             RPTR = COPY_VARIANT_LIST(.RPTR)
```

```
;  4976          5087  3
;  4977          5088  3          ELSE
;  4978          5089  4              BEGIN
;  4979          5090  4              RPTR[RST$B_KIND] = RST$K_TYPCOMP;
;  4980          5091  4              RPTR[RST$L_UPSCOPEPTR] = .TYPRSTPTR;
;  4981          5092  3              END;
;  4982          5093  3
;  4983          5094  3          NEWPTR[.I] = .RPTR;
;  4984          5095  2          END;
;  4985          5096  2
;  4986          5097  2
;  4987          5098  2      ! Finally release the old component list back to the free memory pool and
;  4988          5099  2      ! return to the caller.
;  4989          5100  2      !
;  4990          5101  2      DBG$REL_MEMORY(.OLDPTR);
;  4991          5102  2      RETURN;
;  4992          5103  2
;  4993          5104  1      END;


                                001C 00000 COPY_COMP_LIST:
                                           .WORD   Save R2,R3,R4                                   ; 5030
                54 00000000'   EF  9E 00002        MOVAB   TYPRSTLOC, R4                           ;
                53             64  D0 00009        MOVL    TYPRSTLOC, NEWPTR                       ; 5078
                50       04    BC  D0 0000C        MOVL    @OLDPTR, R0                             ; 5079
                74         9440  DE 00010        MOVAL   @TYPRSTLOC[R0], TYPRSTLOC               ;
                64           04  C0 00014        ADDL2   #4, TYPRSTLOC                           ;
                63       C4  BC  D0 00017        MOVL    @OLDPTR, (NEWPTR)                       ; 5080
                             52  D4 0001B        CLRL    I                                       ; 5081
                             21  11 0001D        BRB     4$                                      ;
                         50      04 BC42 D0 0001F 1$:    MOVL    @OLDPTR[I], RPTR                ; 5083
                         0B      14  A0 91 00024        CMPB    20(RPTR), #11                   ; 5084
                                     09 12 00028        BNEQ    2$                              ;
                                     50 DD 0002A        PUSHL   RPTR                            ; 5086
                       0000V CF                 01 FB 0002C        CALLS   #1, COPY_VARIANT_LIST   ;
                                     09 11 00031        BRB     3$                              ;
                         14  A0                 0A 90 00033 2$:    MOVB    #10, 20(RPTR)           ; 5090
                         10  A0      04  A4 D0 00037        MOVL    TYPRSTPTR, 16(RPTR)             ; 5091
                           6342              50 D0 0003C 3$:    MOVL    RPTR, (NEWPTR)[I]           ; 5094
                DA           52      04  BC F3 00040 4$:    AOBLEQ  @OLDPTR, I, 1$                  ; 5081
                                     04  AC DD 00045        PUSHL   OLDPTR                          ; 5101
                   00000000G 00                 01 FB 00048        CALLS   #1, DBG$REL_MEMORY      ;
                                                04 0004F        RET                                 ; 5104

; Routine Size:  80 bytes,    Routine Base:  DBG$CODE + 2793
```

```
;  4995          5105  1 ROUTINE COPY_VARIANT_LIST(OLDRSTPTR) =
;  4996          5106  1 !
;  4997          5107  1 ! FUNCTION
;  4998          5108  1 !       This routine copies a Variant RST Entry and the corresponding variant
;  4999          5109  1 !       set list from free-standing memory blocks into memory inside the current
;  5000          5110  1 !       Data Type RST Entry's memory block.  This is done so that the variant
;  5001          5111  1 !       set list can be grown dynamically while being accumulated, and still use
;  5002          5112  1 !       only just enough space in the final RST entry.
;  5003          5113  1 !
;  5004          5114  1 !       This routine is called only by COPY_COMP_LIST, which it also calls re-
;  5005          5115  1 !       cursively.
;  5006          5116  1 !
;  5007          5117  1 ! INPUTS
;  5008          5118  1 !       OLDRSTPTR - A pointer to the Variant RST Entry to be copied into the
;  5009          5119  1 !                   Data Type RST Entry memory block.  This "old" RST entry and
;  5010          5120  1 !                   the associated component list are released back to the free
;  5011          5121  1 !                   memory pool.
;  5012          5122  1 !
;  5013          5123  1 ! OUTPUTS
;  5014          5124  1 !       A pointer to the new copy of the Variant RST Entry is returned as the
;  5015          5125  1 !                   routine value.  This copy is located within the current
;  5016          5126  1 !                   Data Type RST Entry.
;  5017          5127  1 !
;  5018          5128  1
;  5019          5129  2     BEGIN
;  5020          5130  2
;  5021          5131  2     MAP
;  5022          5132  2         OLDRSTPTR: REF RST$ENTRY;       ! Pointer to Variant RST Entry to copy
;  5023          5133  2
;  5024          5134  2     LOCAL
;  5025          5135  2         NEWPTR: REF VECTOR[,LONG],      ! Pointer to new variant set table
;  5026          5136  2         NEWRSTPTR: REF RST$ENTRY,       ! Pointer to the new Variant RST Entry
;  5027          5137  2         OLDPTR: REF VECTOR[,LONG];      ! Pointer to "old" variant set list
;  5028          5138  2
;  5029          5139  2
;  5030          5140  2
;  5031          5141  2     ! Allocate space in the current Data Type RST Entry for the new Variant RST
;  5032          5142  2     ! Entry and its component list.
;  5033          5143  2     !
;  5034          5144  2     OLDPTR = .OLDRSTPTR[RST$L_SYMCHNPTR];
;  5035          5145  2     NEWRSTPTR = .TYPRSTLOC;
;  5036          5146  2     TYPRSTLOC = TYPRSTLOC[RST$K_VARENTSIZ + .OLDPTR[0]/2];
;  5037          5147  2     NEWPTR = NEWRSTPTR[RST$A_VARSETTBL];
;  5038          5148  2
;  5039          5149  2
;  5040          5150  2     ! Fill in the fields in the Variant RST Entry.
;  5041          5151  2     !
;  5042          5152  2     NEWRSTPTR[RST$L_HASH_FLINK] = .NEWRSTPTR;
;  5043          5153  2     NEWRSTPTR[RST$L_HASH_BLINK] = .NEWRSTPTR;
;  5044          5154  2     NEWRSTPTR[RST$L_SYMCHNPTR] = 0;
;  5045          5155  2     NEWRSTPTR[RST$L_DSTPTR] = .OLDRSTPTR[RST$L_DSTPTR];
;  5046          5156  2     NEWRSTPTR[RST$L_VARTAGPTR] = .OLDRSTPTR[RST$L_VARTAGPTR];
;  5047          5157  2     NEWRSTPTR[RST$B_KIND] = RST$K_VARIANT;
;  5048          5158  2     NEWRSTPTR[RST$W_REFCOUNT] = 0;
;  5049          5159  2
;  5050          5160  2
;  5051          5161  2     ! Copy the variant set component list (which points to the tag value DST
```

```
;  5052          5162  2     ! record and the variant component list for each variant in the set) over
;  5053          5163  2     ! to the new Variant RST Entry in the current Type RST Entry's memory
;  5054          5164  2     ! block.  Note that we call COPY_COMP_LIST which may recursively call this
;  5055          5165  2     ! routine again.
;  5056          5166  2     !
;  5057          5167  2     NEWRSTPTR[RST$L_VARSET_CNT] = .OLDPTR[0]/2;
;  5058          5168  2     INCR I FROM 1 TO .OLDPTR[0]/2 DO
;  5059          5169  3         BEGIN
;  5060          5170  3         NEWPTR[.I - 1] = .TYPRSTLOC;
;  5061          5171  3         TYPRSTLOC[0] = .OLDPTR[2*.I - 1];
;  5062          5172  3         TYPRSTLOC = TYPRSTLOC[1];
;  5063          5173  3         COPY_COMP_LIST(.OLDPTR[2*.I]);
;  5064          5174  2         END;
;  5065          5175  2
;  5066          5176  2
;  5067          5177  2     ! Finally release the old variant RST entry and component list memory blocks
;  5068          5178  2     ! back to the memory pool.  Return the address of the new Variant RST Entry
;  5069          5179  2     ! to the caller.
;  5070          5180  2     !
;  5071          5181  2     DBG$REL_MEMORY(.OLDRSTPTR);
;  5072          5182  2     DBG$REL_MEMORY(.OLDPTR);
;  5073          5183  2     RETURN .NEWRSTPTR;
;  5074          5184  2
;  5075          5185  1     END;


                                        03FC 00000 COPY_VARIANT_LIST:
                                                   .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9                ; 5105
                59 00000000G   00   9E 00002       MOVAB   DBG$REL_MEMORY, R9
                58 00000000'   EF   9E 00009       MOVAB   TYPRSTLOC, R8
                57        04   AC   D0 00010       MOVL    OLDRSTPTR, R7                               ; 5144
                55        08   A7   D0 00014       MOVL    8(R7), OLDPTR
                53             68   D0 00018       MOVL    TYPRSTLOC, NEWRSTPTR                        ; 5145
56              65             02   C7 0001B       DIVL3   #2, (OLDPTR), R6                            ; 5146
                78           9846   DE 0001F       MOVAL   @TYPRSTLOC[R6], TYPRSTLOC
                68             18   C0 00023       ADDL2   #24, TYPRSTLOC
                54        18   A3   9E 00026       MOVAB   24(R3), NEWPTR                              ; 5147
                63             53   D0 0002A       MOVL    NEWRSTPTR, (NEWRSTPTR)                      ; 5152
          04    A3             53   D0 0002D       MOVL    NEWRSTPTR, 4(NEWRSTPTR)                     ; 5153
                          08   A3   D4 00031       CLRL    8(NEWRSTPTR)                                ; 5154
          0C    A3        0C   A7   7D 00034       MOVQ    12(R7), 12(NEWRSTPTR)                       ; 5155
          14    A3             08   90 00039       MOVB    #11, 20(NEWRSTPTR)                          ; 5157
                          16   A3   B4 0003D       CLRW    22(NEWRSTPTR)                               ; 5158
          08    A3             56   D0 00040       MOVL    R6, 8(NEWRSTPTR)                            ; 5167
                               52   D4 00044       CLRL    I                                           ; 5170
                               1E   11 00046       BRB     2$
          FC A442              68   D0 00048 1$:   MOVL    TYPRSTLOC, -4(NEWPTR)[I]
50              52             01   78 0004D       ASHL    #1, I, R0                                   ; 5171
          00    B8        FC A540   D0 00051       MOVL    -4(OLDPTR)[R0], @TYPRSTLOC
                68             04   C0 00057       ADDL2   #4, TYPRSTLOC                               ; 5172
50              52             01   78 0005A       ASHL    #1, I, R0                                   ; 5173
                             6540   DD 0005E       PUSHL   (OLDPTR)[R0]
          FF4A  CF             01   FB 00061       CALLS   #1, COPY_COMP_LIST
DE              52             56   F3 00066 2$:   AOBLEQ  R6, I, 1$                                  ; 5168
```

```
                     57  DD 0006A        PUSHL   R7                              ; 5181
                69   01  FB 0006C        CALLS   #1, DBG$REL_MEMORY              ;
                     55  DD 0006F        PUSHL   OLDPTR                          ; 5182
                69   01  FB 00071        CALLS   #1, DBG$REL_MEMORY              ;
                50   53  D0 00074        MOVL    NEWRSTPTR, R0                   ; 5183
                         04 00077        RET                                     ; 5185

; Routine Size:  120 bytes,    Routine Base:  DBG$CODE + 27E3
```

```
;  5077          5186  1 ROUTINE COMP_LIST_ADD_ENTRY(COMP_LST_PTR, COMP_LST_SIZE, VALUE)  NOVALUE =
;  5078          5187  1 !
;  5079          5188  1 ! FUNCTION
;  5080          5189  1 !       This routine adds an entry to a "component list".  Such a list consists
;  5081          5190  1 !       of a memory block whose first longword contains a count giving the num-
;  5082          5191  1 !       ber of entries on the list.  The longwords that follow contain the list
;  5083          5192  1 !       elements, one per longword.  This routine increments the count and ap-
;  5084          5193  1 !       pends the new component value to the list.  If this would cause the list
;  5085          5194  1 !       memory block to overflow, a new, larger block is allocated and the list
;  5086          5195  1 !       is copied to that block before the new value is appended.
;  5087          5196  1 !
;  5088          5197  1 ! INPUTS
;  5089          5198  1 !       COMP_LST_PTR - The address of the pointer to the component list memory
;  5090          5199  1 !                      block to which the new value should be appended.
;  5091          5200  1 !
;  5092          5201  1 !       COMP_LST_SIZE - The address of the location giving the number of list
;  5093          5202  1 !                       components the memory block can hold.  This is one less than
;  5094          5203  1 !                       the size of the list memory block in longwords.
;  5095          5204  1 !
;  5096          5205  1 !       VALUE   - The value to be appended at the end of the component list.
;  5097          5206  1 !
;  5098          5207  1 ! OUTPUTS
;  5099          5208  1 !       COMP_LST_PTR - If a new memory block had to be allocated to accommodate
;  5100          5209  1 !                      the new list entry, the address of that new memory block is
;  5101          5210  1 !                      returned to COMP_LST_PTR.  The old block is deallocated.
;  5102          5211  1 !
;  5103          5212  1 !       COMP_LST_SIZE - If a new memory block had to be allocated to accommodate
;  5104          5213  1 !                       the new list entry, the number of list entries the new memory
;  5105          5214  1 !                       block can hold is returned to COMP_LST_SIZE.
;  5106          5215  1 !
;  5107          5216  1
;  5108          5217  2     BEGIN
;  5109          5218  2
;  5110          5219  2     MAP
;  5111          5220  2         COMP_LST_PTR: REF VECTOR[1],    ! Pointer to pointer to component list
;  5112          5221  2         COMP_LST_SIZE: REF VECTOR[1];   ! Pointer to size of component list
;  5113          5222  2
;  5114          5223  2     LOCAL
;  5115          5224  2         LISTPTR: REF VECTOR[,LONG],     ! Pointer to component list
;  5116          5225  2         NEWPTR;                         ! Pointer to new memory block for list
;  5117          5226  2
;  5118          5227  2
;  5119          5228  2
;  5120          5229  2     ! If the component list memory block is about to overflow, get a new,
;  5121          5230  2     ! larger memory block and copy the contents of the old block into the
;  5122          5231  2     ! new block.  Then release the old block to the free memory pool.
;  5123          5232  2     !
;  5124          5233  2     LISTPTR = .COMP_LST_PTR[0];
;  5125          5234  2     IF .LISTPTR[0] EQL .COMP_LST_SIZE[0]
;  5126          5235  2     THEN
;  5127          5236  3         BEGIN
;  5128          5237  3         NEWPTR = DBG$GET_MEMORY(.COMP_LST_SIZE[0] + 11);
;  5129          5238  3         CH$MOVE(4*(.LISTPTR[0] + 1), .LISTPTR, .NEWPTR);
;  5130          5239  3         COMP_LST_PTR[0] = .NEWPTR;
;  5131          5240  3         COMP_LST_SIZE[0] = .COMP_LST_SIZE[0] + 10;
;  5132          5241  3         DBG$REL_MEMORY(.LISTPTR);
;  5133          5242  3         LISTPTR = .NEWPTR;
```

```
;  5134     5243  2          END;
;  5135     5244  2
;  5136     5245  2
;  5137     5246  2      ! Append the new value at the end of the component list vector.
;  5138     5247  2      !
;  5139     5248  2      LISTPTR[0] = .LISTPTR[0] + 1;
;  5140     5249  2      LISTPTR[.LISTPTR[0]] = .VALUE;
;  5141     5250  2      RETURN;
;  5142     5251  2
;  5143     5252  1      END;


                                                00FC 00000 COMP_LIST_ADD_ENTRY:
                                                           .WORD   Save R2,R3,R4,R5,R6,R7                  ; 5186
                        56        04    BC  D0 00002         MOVL    @COMP_LST_PTR, LISTPTR                 ; 5233
                        52              66  D0 00006         MOVL    (LISTPTR), R2                          ; 5234
                   08   BC              52  D1 00009         CMPL    R2, @COMP_LST_SIZE
                                        2E  12 0000D         BNEQ    1$
                7E      08   BC         0B  C1 0000F         ADDL3   #11, @COMP_LST_SIZE, -(SP)             ; 5237
            00000000G   00              01  FB 00014         CALLS   #1, DBG$GET_MEMORY
                        57              50  D0 0001B         MOVL    R0, NEWPTR
                50      52              02  78 0001E         ASHL    #2, R2, R0                             ; 5238
                        50              04  C0 00022         ADDL2   #4, R0
                67      66              50  28 00025         MOVC3   R0, (LISTPTR), (NEWPTR)
                   04   BC              57  D0 00029         MOVL    NEWPTR, @COMP_LST_PTR                  ; 5239
                   08   BC              0A  C0 0002D         ADDL2   #10, @COMP_LST_SIZE                    ; 5240
                                        56  DD 00031         PUSHL   LISTPTR                                ; 5241
            00000000G   00              01  FB 00033         CALLS   #1, DBG$REL_MEMORY
                        56              57  D0 0003A         MOVL    NEWPTR, LISTPTR                        ; 5242
                                        66  D6 0003D 1$:     INCL    (LISTPTR)                              ; 5248
                        50              66  D0 0003F         MOVL    (LISTPTR), R0                          ; 5249
                      6640      0C      AC  D0 00042         MOVL    VALUE, (LISTPTR)[R0]
                                            04 00047         RET                                            ; 5252

; Routine Size:  72 bytes,    Routine Base:  DBG$CODE + 285B
```

```
;  5145          5253  1 ROUTINE GET_MEM(LENGTH) =
;  5146          5254  1 !
;  5147          5255  1 ! FUNCTION
;  5148          5256  1 !       This routine gets a memory block of a specified size from the free mem-
;  5149          5257  1 !       ory pool.  If there is no memory left in the pool, it expands the free
;  5150          5258  1 !       memory pool via DBG$EXPAND_MEMORY (which gets more space from the system
;  5151          5259  1 !       via $EXPREG).  It then gets the desired block from the expanded pool.
;  5152          5260  1 !       If the expansion fails for some reason, zero is returned instead of a
;  5153          5261  1 !       block address.
;  5154          5262  1 !
;  5155          5263  1 !       This routine is used only during Run-Time Symbol Table initialization.
;  5156          5264  1 !       After that, the free memory pool is not allowed to expand into the user
;  5157          5265  1 !       address space because that could cause DEBUG space and user space to
;  5158          5266  1 !       become checkerboarded.  That would make it more likely that the Debugger
;  5159          5267  1 !       could interfere with the behavior of the user program being debugged.
;  5160          5268  1 !
;  5161          5269  1 ! INPUTS
;  5162          5270  1 !       LENGTH  - The length in longwords of the desired memory block.
;  5163          5271  1 !
;  5164          5272  1 ! OUTPUTS
;  5165          5273  1 !       The start address of the returned memory block is returned as the value
;  5166          5274  1 !       of GET_MEM.  If no memory could be found, an error is signalled.
;  5167          5275  1 !
;  5168          5276  1
;  5169          5277  2     BEGIN
;  5170          5278  2
;  5171          5279  2     LOCAL
;  5172          5280  2         ADDRESS,                        ! Pointer to the allocated memory block
;  5173          5281  2         STATUS: BLOCK[1,LONG];          ! Status code from DBG$EXPAND_MEMORY
;  5174          5282  2
;  5175          5283  2
;  5176          5284  2
;  5177          5285  2     ! Get the desired memory block from DBG$GET_MEMORY.  If that succeeds,
;  5178          5286  2     ! simply return the block's address right away.
;  5179          5287  2     !
;  5180          5288  2     ADDRESS = DBG$GET_MEMORY(.LENGTH, 0);
;  5181          5289  2     IF .ADDRESS NEQ 0 THEN RETURN .ADDRESS;
;  5182          5290  2
;  5183          5291  2
;  5184          5292  2     ! If we could not get any memory, try to expand the memory pool.  If that
;  5185          5293  2     ! fails, signal an error message (no free memory).
;  5186          5294  2     !
;  5187          5295  2     STATUS = DBG$EXPAND_MEMORY(MAX(16384 - 3, .LENGTH + 1));
;  5188          5296  2     IF NOT .STATUS THEN SIGNAL(DBG$_NOFREE);
;  5189          5297  2
;  5190          5298  2
;  5191          5299  2     ! The memory pool was successfully expanded.  Now allocate the block we
;  5192          5300  2     ! desire--this time it is guaranteed to work.
;  5193          5301  2     !
;  5194          5302  2     RETURN DBG$GET_MEMORY(.LENGTH);
;  5195          5303  2
;  5196          5304  1     END;
```

```
                                                0004 00000 GET_MEM:.WORD    Save R2                                 ; 5253
                                52 00000000G    00   9E 00002           MOVAB   DBG$GET_MEMORY, R2
                                                7E   D4 00009           CLRL    -(SP)                           ; 5288
                                         04     AC   DD 0000B           PUSHL   LENGTH
                                62              02   FB 0000E           CALLS   #2, DBG$GET_MEMORY
                                                50   D5 00011           TSTL    ADDRESS                         ; 5289
                                                32   12 00013           BNEQ    3$
                        50              04      AC   01   C1 00015       ADDL3   #1, LENGTH, R0                  ; 5295
                                                50   DD 0001A           PUSHL   R0
                            00003FFD    8F      6E   D1 0001C           CMPL    (SP), #16381
                                                05   18 00023           BGEQ    1$
                                6E          3FFD    8F   3C 00025       MOVZWL  #16381, (SP)
                            00000000G       00      01   FB 0002A 1$:   CALLS   #1, DBG$EXPAND_MEMORY
                                            0D      50   E8 00031       BLBS    STATUS, 2$                      ; 5296
                                        00028332    8F   DD 00034       PUSHL   #164658
                            00000000G       00      01   FB 0003A       CALLS   #1, LIB$SIGNAL
                                             04     AC   DD 00041 2$:    PUSHL   LENGTH                          ; 5302
                                62                  01   FB 00044       CALLS   #1, DBG$GET_MEMORY
                                                    04   00047 3$:      RET                                     ; 5304

; Routine Size:  72 bytes,    Routine Base:  DBG$CODE + 28A3
```

```
;  5198          5305  1 ROUTINE GET_MOD_RST(MODNAMEPTR, LENGTH) =
;  5199          5306  1 !
;  5200          5307  1 ! FUNCTION
;  5201          5308  1 !       This routine looks up a specified module name in the RST and returns a
;  5202          5309  1 !       pointer to the corresponding Module RST Entry.
;  5203          5310  1 !
;  5204          5311  1 ! INPUTS
;  5205          5312  1 !       MODNAMEPTR - A pointer to the first character of the ASCII string con-
;  5206          5313  1 !                    taining the desired module's name.
;  5207          5314  1 !
;  5208          5315  1 !       LENGTH  - The length of the module name in characters.
;  5209          5316  1 !
;  5210          5317  1 ! OUTPUTS
;  5211          5318  1 !       A pointer to the specified module's Module RST Entry is returned as the
;  5212          5319  1 !       routine's value.  If no such routine exists in the RST, a zero
;  5213          5320  1 !       is returned instead.
;  5214          5321  1 !
;  5215          5322  1
;  5216          5323  2     BEGIN
;  5217          5324  2
;  5218          5325  2     LOCAL
;  5219          5326  2         DSTNAME: REF VECTOR[,BYTE],     ! Pointer to the current module's name
;  5220          5327  2                                         !      in the DST (as Counted ASCII)
;  5221          5328  2         RSTPTR: REF RST$ENTRY;          ! A pointer to the current Module RST
;  5222          5329  2                                         !      Entry
;  5223          5330  2
;  5224          5331  2
;  5225          5332  2
;  5226          5333  2     ! Search the RST Module Chain for a module with the specified name.  If we
;  5227          5334  2     ! find such a module, exit the search loop with RSTPTR pointing to the
;  5228          5335  2     ! desired Module RST Entry.  If we exhaust the loop, RSTPTR is zero.  In
;  5229          5336  2     ! either case, return whatever value RSTPTR has.
;  5230          5337  2     !
;  5231          5338  2     RSTPTR = .RST$START_ADDR;
;  5232          5339  2     WHILE .RSTPTR NEQ 0 DO
;  5233          5340  3         BEGIN
;  5234          5341  3         IF NOT .RSTPTR[RST$V_ANONMOD]
;  5235          5342  3         THEN
;  5236          5343  4             BEGIN
;  5237          5344  4             DSTNAME = DBG$GET_DST_NAME(.RSTPTR[RST$L_DSTPTR]);
;  5238          5345  4             IF CH$EQL(.DSTNAME[0], DSTNAME[1], .LENGTH, .MODNAMEPTR, 0)
;  5239          5346  4             THEN
;  5240          5347  4                 EXITLOOP;
;  5241          5348  4
;  5242          5349  3             END;
;  5243          5350  3
;  5244          5351  3         RSTPTR = .RSTPTR[RST$L_NXTMODPTR];
;  5245          5352  2         END;
;  5246          5353  2
;  5247          5354  2     RETURN .RSTPTR;
;  5248          5355  2
;  5249          5356  1     END;
```

```
                                                003C 00000 GET_MOD_RST:
                                                           .WORD   Save R2,R3,R4,R5                                        ; 5305
                                        54 00000000'    EF  D0 00002            MOVL    RST$START_ADDR, RSTPTR                  ; 5338
                                                        24  13 00009 1$:        BEQL    3$                                      ; 5339
                                    19      28      A4      02  E0 0000B        BBS     #2, 40(RSTPTR), 2$                      ; 5341
                                                    0C  A4  DD 00010            PUSHL   12(RSTPTR)                              ; 5344
                                            D768    CF      01  FB 00013        CALLS   #1, DBG$GET_DST_NAME                    ;
                                                    55      50  D0 00018        MOVL    R0, DSTNAME                             ;
                                                    50      65  9A 0001B        MOVZBL  (DSTNAME), R0                           ; 5345
    08  AC              00      01      A5          50  2D 0001E                CMPC5   R0, 1(DSTNAME), #0, LENGTH, @MODNAMEPTR ;
                                                04  BC      00025               ;
                                                    06  13 00027              BEQL    3$                                      ;
                                        54      10  A4  D0 00029 2$:          MOVL    16(RSTPTR), RSTPTR                      ; 5351
                                                    DA  11 0002D              BRB     1$                                      ; 5339
                                        50          54  D0 0002F 3$:          MOVL    RSTPTR, R0                              ; 5354
                                                        04 00032              RET                                             ; 5356

; Routine Size:  51 bytes,    Routine Base:  DBG$CODE + 28EB
```

```
;  5251          5357  1 ROUTINE GET_TYPEPTR(DATA_DSTPTR, MODNAMEPTR, TYPE_DSTPTR, ARRAY_FLAG): NOVALUE =
;  5252          5358  1 !
;  5253          5359  1 ! FUNCTION
;  5254          5360  1 !       This routine determines whether a given data item is of a record type or
;  5255          5361  1 !       an enumeration type, and if so, what the address is of the DST record
;  5256          5362  1 !       which defines that type (the Record Begin or Enumeration Type Begin DST
;  5257          5363  1 !       record).  The routine is called during the RST build procedure so that
;  5258          5364  1 !       the RST$L_TYPEPTR field can be filled in for record and enumeration type
;  5259          5365  1 !       data items.
;  5260          5366  1 !
;  5261          5367  1 !       If the given data item is of an array type, this routine determines if
;  5262          5368  1 !       the array's cell type is a record data type.  If so, the address of the
;  5263          5369  1 !       cell type's Record Begin DST record is returned and TRUE is returned to
;  5264          5370  1 !       the ARRAY_FLAG parameter.  This information allows the RST build proce-
;  5265          5371  1 !       dure to build the record type's Type RST Entry so that it has a refer-
;  5266          5372  1 !       ence pointer to the array's Data RST Entry.  This information is later
;  5267          5373  1 !       used by the DBG$STA_GETSYMBOL routine during symbol lookup.
;  5268          5374  1 !
;  5269          5375  1 ! INPUTS
;  5270          5376  1 !       DATA_DSTPTR - A pointer to the data item DST record whose type is to be
;  5271          5377  1 !                     determined.
;  5272          5378  1 !
;  5273          5379  1 !       MODNAMEPTR - A pointer to the current module's name as Counted ASCII.
;  5274          5380  1 !                    This is used only for error messages.
;  5275          5381  1 !
;  5276          5382  1 !       TYPE_DSTPTR - A pointer to a longword location to receive a pointer
;  5277          5383  1 !                     to a Record Begin or Enumeration Type Begin DST record.
;  5278          5384  1 !
;  5279          5385  1 !       ARRAY_FLAG - A pointer to a longword location to receive the array
;  5280          5386  1 !                    flag returned by this routine.
;  5281          5387  1 !
;  5282          5388  1 ! OUTPUTS
;  5283          5389  1 !       TYPE_DSTPTR - If the data item is an object of a record or enumeration
;  5284          5390  1 !                     data type, the address of the corresponding Record Begin or
;  5285          5391  1 !                     Enumeration Type Begin DST record is returned to TYPE_DSTPTR.
;  5286          5392  1 !                     If the data item is an array whose cell type is a record data
;  5287          5393  1 !                     type, the address of the cell type's Record Begin DST record
;  5288          5394  1 !                     is returned to TYPE_DSTPTR.  In all other cases, zero is re-
;  5289          5395  1 !                     turned to TYPE_DSTPTR.
;  5290          5396  1 !
;  5291          5397  1 !       ARRAY_FLAG - If the data item is an array whose cell type is a record
;  5292          5398  1 !                    data type, TRUE is returned to ARRAY_FLAG.  Otherwise, FALSE
;  5293          5399  1 !                    is returned to ARRAY_FLAG.
;  5294          5400  1 !
;  5295          5401  1 !       No value is returned.
;  5296          5402  1 !
;  5297          5403  1
;  5298          5404  2     BEGIN
;  5299          5405  2
;  5300          5406  2     MAP
;  5301          5407  2         TYPE_DSTPTR: REF VECTOR[1],     ! Pointer to Type DST pointer longword
;  5302          5408  2         ARRAY_FLAG: REF VECTOR[1];      ! Pointer to array flag longword
;  5303          5409  2
;  5304          5410  2     LOCAL
;  5305          5411  2         DSTPTR: REF DST$RECORD,         ! Pointer to the current DST record
;  5306          5412  2         FBPTR: REF BITVECTOR[],         ! Pointer to Array Type Spec flag bits
;  5307          5413  2         LOCAL_ARRAY_FLAG,               ! Set to TRUE if data item is an array
```

```
;  5308          5414  2          TSPTR: REF DST$TYPE_SPEC,       ! Pointer to a DST Type Spec
;  5309          5415  2          VSPTR: REF DST$VAL_SPEC;        ! Pointer to the descriptor Value Spec
;  5310          5416  2                                          !       in an Array Type Spec
;  5311          5417  2
;  5312          5418  2
;  5313          5419  2
;  5314          5420  2      ! Initially we clear both output parameters.
;  5315          5421  2      !
;  5316          5422  2      TYPE_DSTPTR[0] = 0;
;  5317          5423  2      ARRAY_FLAG[0] = FALSE;
;  5318          5424  2
;  5319          5425  2
;  5320          5426  2      ! Unless the data item is a DST$K_SEPTYP DST record, it cannot point to a
;  5321          5427  2      ! record or enumeration type definition, so we return immediately.
;  5322          5428  2      !
;  5323          5429  2      DSTPTR = .DATA_DSTPTR;
;  5324          5430  2      IF .DSTPTR[DST$B_TYPE] NEQ DST$K_SEPTYP THEN RETURN;
;  5325          5431  2
;  5326          5432  2
;  5327          5433  2      ! Advance DSTPTR past the DST$K_SEPTYP record.  If the next record is a
;  5328          5434  2      ! Global-is-Next record, skip past that too to get to the record
;  5329          5435  2      ! defining the data type.
;  5330          5436  2      !
;  5331          5437  2  !       DSTPTR = DSTPTR[DST$A_NEXT] + .DSTPTR[DST$B_LENGTH];                    ! D015
;  5332          5438  2      DSTPTR = DBG$RST_NEXT_DST( .DSTPTR );
;  5333          5439  2      IF .DSTPTR[DST$B_TYPE] EQL DST$K_GLOBNXT
;  5334          5440  2      THEN
;  5335          5441  2  !       DSTPTR = DSTPTR[DST$A_NEXT] + .DSTPTR[DST$B_LENGTH];                    ! D015
;  5336          5442  2          DSTPTR = DBG$RST_NEXT_DST( .DSTPTR );
;  5337          5443  2
;  5338          5444  2
;  5339          5445  2      ! Loop through any Type Spec indirection in the DST so we can get to the
;  5340          5446  2      ! ultimate type specification for the data item.
;  5341          5447  2      !
;  5342          5448  2      LOCAL_ARRAY_FLAG = FALSE;
;  5343          5449  2      WHILE TRUE DO
;  5344          5450  3          BEGIN
;  5345          5451  3
;  5346          5452  3
;  5347          5453  3          ! If we have now gotten to a Record Begin or Enumeration Type Begin DST
;  5348          5454  3          ! record, we exit the indirection loop and return a pointer to that DST
;  5349          5455  3          ! record to TYPE_DSTPTR.
;  5350          5456  3          !
;  5351          5457  3          IF .DSTPTR[DST$B_TYPE] EQL DST$K_RECBEG THEN EXITLOOP;
;  5352          5458  3          IF .DSTPTR[DST$B_TYPE] EQL DST$K_ENUMBEG
;  5353          5459  3          THEN
;  5354          5460  4              BEGIN
;  5355          5461  4              IF .LOCAL_ARRAY_FLAG THEN RETURN;
;  5356          5462  4              EXITLOOP;
;  5357          5463  4              END;
;  5358          5464  3
;  5359          5465  3
;  5360          5466  3          ! If DSTPTR points to anything other than a Record Begin, Enumeration
;  5361          5467  3          ! Type Begin, or Type Spec DST record, there is an error in the DST.
;  5362          5468  3          !
;  5363          5469  3          IF .DSTPTR[DST$B_TYPE] NEQ DST$K_TYPSPEC
;  5364          5470  3          THEN
```

```
;  5365          5471  3            SIGNAL(DBG$_INCDSTNES, 1, .MODNAMEPTR);
;  5366          5472  3
;  5367          5473  3
;  5368          5474  3        ! This is a Type Spec DST record.  Loop through any nested cell types
;  5369          5475  3        ! in Array Type Specs until we get to an Indirect Type Spec or a Novel
;  5370          5476  3        ! Length Type Spec (which also indirects).  If we get to an Indirect
;  5371          5477  3        ! or Novel Length Type Spec, we pick up the new DST pointer and con-
;  5372          5478  3        ! tinue the outer loop; otherwise we return a zero to TYPE_DSTPTR.
;  5373          5479  3        !
;  5374          5480  3        TSPTR = DSTPTR[DST$A_TYPSPEC_TS_ADDR] + .DSTPTR[DST$B_TYPSPEC_NAME];
;  5375          5481  3        WHILE TRUE DO
;  5376          5482  4            BEGIN
;  5377          5483  4
;  5378          5484  4
;  5379          5485  4            ! If this is an Indirect Type Spec, get the new DST pointer and
;  5380          5486  4            ! continue the outer loop.
;  5381          5487  4            !
;  5382          5488  4            IF .TSPTR[DST$B_TS_KIND] EQL DST$K_TS_IND
;  5383          5489  4            THEN
;  5384          5490  5                BEGIN
;  5385          5491  5                DSTPTR = DBG$RST_DST_PTR( .TSPTR[DST$L_TS_IND_PTR] + .DST$BEGIN_ADDR );
;  5386          5492  5                EXITLOOP;
;  5387          5493  5                END
;  5388          5494  5
;  5389          5495  5
;  5390          5496  5            ! If this is a Novel Length Type Spec, do the same thing.
;  5391          5497  5            !
;  5392          5498  4            ELSE IF .TSPTR[DST$B_TS_KIND] EQL DST$K_TS_NOV_LENG
;  5393          5499  4            THEN
;  5394          5500  5                BEGIN
;  5395          5501  5                DSTPTR = DBG$RST_DST_PTR( .TSPTR[DST$L_TS_NOV_LENG_PAR_TSPEC] + .DST$BEGIN_ADDR );
;  5396          5502  5                EXITLOOP;
;  5397          5503  5                END
;  5398          5504  5
;  5399          5505  5
;  5400          5506  5            ! If this is an Array Type Spec, set the array flag and determine
;  5401          5507  5            ! whether the cell type can be a record type.  If not, return zero
;  5402          5508  5            ! to TYPE_DSTPTR.  Otherwise, get the pointer to the cell type Type
;  5403          5509  5            ! Spec and continue the inner loop.
;  5404          5510  5            !
;  5405          5511  4            ELSE IF .TSPTR[DST$B_TS_KIND] EQL DST$K_TS_ARRAY
;  5406          5512  4            THEN
;  5407          5513  5                BEGIN
;  5408          5514  5                LOCAL_ARRAY_FLAG = TRUE;
;  5409          5515  5                FBPTR = TSPTR[DST$A_TS_ARRAY_FLAGS_ADDR];
;  5410          5516  5                IF NOT .FBPTR[0] THEN RETURN;
;  5411          5517  5                VSPTR = .FBPTR + (.TSPTR[DST$B_TS_ARRAY_DIM] + 8)/8;
;  5412          5518  5                IF .VSPTR[DST$B_VS_VFLAGS] EQL DST$K_VS_FOLLOWS
;  5413          5519  5                THEN
;  5414          5520  5                    TSPTR = VSPTR[DST$B_VS_ALLOC] + .VSPTR[DST$W_VS_LENGTH]
;  5415          5521  5
;  5416          5522  5                ELSE
;  5417          5523  5                    TSPTR = VSPTR[DST$A_VS_TVS_BASE];
;  5418          5524  5
;  5419          5525  5                END
;  5420          5526  5
;  5421          5527  5
```

```
; 5422      5528  5          ! For any other kind of Type Spec, it is not possible to reach a
; 5423      5529  5          ! Record Begin or Enumeration Type Begin DST record so we return
; 5424      5530  5          ! zero to TYPE_DSTPTR.
; 5425      5531  5          !
; 5426      5532  4          ELSE
; 5427      5533  4              RETURN;
; 5428      5534  4
; 5429      5535  3          END;                        ! End of inner cell type loop
; 5430      5536  3
; 5431      5537  2      END;                            ! End of outer indirection loop
; 5432      5538  2
; 5433      5539  2
; 5434      5540  2      ! We found a Record or Enumeration Type type specification.  Return a
; 5435      5541  2      ! pointer to the corresponding DST record and return the array flag value.
; 5436      5542  2      !
; 5437      5543  2      TYPE_DSTPTR[0] = .DSTPTR;
; 5438      5544  2      IF .DSTPTR NEQ 0 THEN ARRAY_FLAG[0] = .LOCAL_ARRAY_FLAG;
; 5439      5545  2      RETURN;
; 5440      5546  2
; 5441      5547  1      END;


                                00FC 00000 GET_TYPEPTR:
                                           .WORD   Save R2,R3,R4,R5,R6,R7                  ; 5357
                    57 00000000'   EF  9E 00002    MOVAB   DST$BEGIN_ADDR, R7
                               0C   BC  D4 00009    CLRL    @TYPE_DSTPTR                    ; 5422
                               10   BC  D4 0000C    CLRL    @ARRAY_FLAG                     ; 5423
                            54 04   AC  D0 0000F    MOVL    DATA_DSTPTR, DSTPTR             ; 5429
                         A3 8F  01  A4  91 00013    CMPB    1(DSTPTR), #163                 ; 5430
                                    7A  12 00018    BNEQ    10$
                                    54  DD 0001A    PUSHL   DSTPTR                          ; 5438
                      FAF6  CF      01  FB 0001C    CALLS   #1, DBG$RST_NEXT_DST
                            54      50  D0 00021    MOVL    R0, DSTPTR
                         A0 8F  01  A4  91 00024    CMPB    1(DSTPTR), #160                 ; 5439
                                    0A  12 00029    BNEQ    1$
                                    54  DD 0002B    PUSHL   DSTPTR                          ; 5442
                      FAE5  CF      01  FB 0002D    CALLS   #1, DBG$RST_NEXT_DST
                            54      50  D0 00032    MOVL    R0, DSTPTR
                                    56  D4 00035 1$: CLRL    LOCAL_ARRAY_FLAG                ; 5448
                         AB 8F  01  A4  91 00037 2$: CMPB    1(DSTPTR), #171                 ; 5457
                                    03  12 0003C    BNEQ    3$
                                0084 31  0003E    BRW     12$
                         A5 8F  01  A4  91 00041 3$: CMPB    1(DSTPTR), #165                 ; 5458
                                    04  12 00046    BNEQ    4$
                               7A       56  E9 00048    BLBC    LOCAL_ARRAY_FLAG, 12$       ; 5461
                                        04  0004B    RET
                         AF 8F  01  A4  91 0004C 4$: CMPB    1(DSTPTR), #175                 ; 5469
                                    12  13 00051    BEQL    5$
                                08  AC  DD 00053    PUSHL   MODNAMEPTR                      ; 5471
                                    01  DD 00056    PUSHL   #1
                          0002896A  8F  DD 00058    PUSHL   #166250
                00000000G   00      03  FB 0005E    CALLS   #3, LIB$SIGNAL
                            50  02  A4  9A 00065 5$: MOVZBL  2(DSTPTR), R0                  ; 5480
                            52  03 A044 9E 00069    MOVAB   3(R0)[DSTPTR], TSPTR
```

```
                        03      02  A2  91 0006E 6$:     CMPB    2(TSPTR), #3                                    ; 5488
                                    07  12 00072         BNEQ    7$
                7E      03      A2      67  C1 00074     ADDL3   DST$BEGIN_ADDR, 3(TSPTR), -(SP)               ; 5491
                                    0B  11 00079         BRB     8$
                        0E      02  A2  91 0007B 7$:     CMPB    2(TSPTR), #14                                   ; 5498
                                    0F  12 0007F         BNEQ    9$
                7E      07      A2      67  C1 00081     ADDL3   DST$BEGIN_ADDR, 7(TSPTR), -(SP)               ; 5501
                      F1A3      CF      01  FB 00086 8$:     CALLS   #1, DBG$RST_DST_PTR
                        54              50  D0 0008B     MOVL    R0, DSTPTR
                                        A7  11 0008E     BRB     2$                                              ; 5500
                        07      02  A2  91 00090 9$:     CMPB    2(TSPTR), #7                                    ; 5511
                                    39  12 00094 10$:     BNEQ    13$
                        56              01  D0 00096     MOVL    #1, LOCAL_ARRAY_FLAG                            ; 5514
                        53      04  A2  9E 00099         MOVAB   4(R2), FBPTR                                    ; 5515
                        2F              63  E9 0009D     BLBC    (FBPTR), 13$                                    ; 5516
                        50      03  A2  9A 000A0         MOVZBL  3(TSPTR), R0                                    ; 5517
                        50              08  C0 000A4     ADDL2   #8, R0
                        50              08  C6 000A7     DIVL2   #8, R0
                7E      50              53  C1 000AA     ADDL3   FBPTR, R0, VSPTR
                      FD      8F        65  91 000AE     CMPB    (VSPTR), #253                                   ; 5518
                                    0B  12 000B2         BNEQ    11$
                        50      01  A5  3C 000B4         MOVZWL  1(VSPTR), R0                                    ; 5520
                        52      03 A045  9E 000B8         MOVAB   3(R0)[VSPTR], TSPTR
                                    AF  11 000BD         BRB     6$
                        52      05  A5  9E 000BF 11$:     MOVAB   5(R5), TSPTR                                    ; 5523
                                    A9  11 000C3         BRB     6$                                              ; 5511
                      0C        BC      54  D0 000C5 12$:     MOVL    DSTPTR, @TYPE_DSTPTR                            ; 5543
                                    04  13 000C9         BEQL    13$                                             ; 5544
                      10        BC      56  D0 000CB     MOVL    LOCAL_ARRAY_FLAG, @ARRAY_FLAG
                                        04  000CF 13$:     RET                                                     ; 5547

; Routine Size:  208 bytes,    Routine Base:  DBG$CODE + 291E
```

```
;  5443          5548  1 ROUTINE MERGE_CONTINUATION_DST(PASSED_DSTPTR, PASSED_NEXT_DSTPTR) =
;  5444          5549  1 !
;  5445          5550  1 ! FUNCTION
;  5446          5551  1 !       Given a DST pointer, this routine scans ahead in the DST to see
;  5447          5552  1 !       if there are continuation records following this DST record.
;  5448          5553  1 !       If so, we merge all of these into a single DST record which
;  5449          5554  1 !       is allocated from DEBUG's memory pool.
;  5450          5555  1 !
;  5451          5556  1 !       It returns TRUE if it encountered a continuation record.
;  5452          5557  1 !
;  5453          5558  1 !       The two parameters are in-out parameters in which we return
;  5454          5559  1 !       the virtual address of the merged DST record and the
;  5455          5560  1 !       virtual address of the next DST record. If there were no
;  5456          5561  1 !       continuation records then the address in PASSED_DSTPTR
;  5457          5562  1 !       is left unchanged.
;  5458          5563  1 !
;  5459          5564  1 !       It assumes that it has been given a DST pointer within the
;  5460          5565  1 !       mapped in DST, not a moved DST record.
;  5461          5566  1 !
;  5462          5567  1 ! INPUTS
;  5463          5568  1 !       PASSED_DSTPTR - Pointer to a DST record passed by reference.
;  5464          5569  1 !
;  5465          5570  1 !       PASSED_NEXT_DSTPTR - Output parameter in which we fill in
;  5466          5571  1 !               the address of the next DST record, after reading
;  5467          5572  1 !               past any continuation records.
;  5468          5573  1 !
;  5469          5574  1 ! OUTPUTS
;  5470          5575  1 !       If there are continuation records, a block of memory large
;  5471          5576  1 !       enough to hold the merged record is allocated, and the
;  5472          5577  1 !       appropriate bytes are copied into this block. A pointer to
;  5473          5578  1 !       this newly constructed DST record is returned in the in-out
;  5474          5579  1 !       parameter PASSED_DSTPTR.
;  5475          5580  1 !
;  5476          5581  1 !       If there are not any continuation records the passed DST pointer
;  5477          5582  1 !       is returned in PASSED_DSTPTR.
;  5478          5583  1 !
;  5479          5584  1 !       In any case the next DST pointer is calculated and returned in
;  5480          5585  1 !       PASSED_NEXT_DSTPTR.
;  5481          5586  1 !
;  5482          5587  1 !       The routine value is TRUE or FALSE, for whether any continuation
;  5483          5588  1 !       records were seen.
;  5484          5589  1 !
;  5485          5590  1
;  5486          5591  2     BEGIN
;  5487          5592  2
;  5488          5593  2     MAP
;  5489          5594  2         PASSED_DSTPTR: REF VECTOR[1],     ! Explained in routine header above
;  5490          5595  2         PASSED_NEXT_DSTPTR: REF VECTOR[1];! Explained in routine header above
;  5491          5596  2
;  5492          5597  2     LOCAL
;  5493          5598  2         CONTINUATION_FLAG:                ! Did we see a continuation
;  5494          5599  2                         INITIAL(FALSE),   !      record?
;  5495          5600  2         DSTPTR: REF DST$RECORD,           ! The current DST pointer
;  5496          5601  2         MOVED_DST_ENTRY: REF MDE$RECORD,  ! Pointer to a Moved DST entry
;  5497          5602  2         NEW_DSTPTR,                       ! New DST record under construction
;  5498          5603  2         NEW_LENGTH,                       ! Size of new DST record
;  5499          5604  2         NEXT_DSTPTR: REF DST$RECORD,      ! Pointer to new DST record
```

```
;  5500     5605  2          NEXT_LEN,                       ! Length of next DST record
;  5501     5606  2          TEMP_PTR;                       ! Temporary pointer to new DST record
;  5502     5607  2
;  5503     5608  2
;  5504     5609  2
;  5505     5610  2      ! Because it is easy to be off by +1 or -1 when copying bytes around,
;  5506     5611  2      ! I'll explain the algorithm in detail using pictures.
;  5507     5612  2      !
;  5508     5613  2      ! Here is our situation at the start of the loop:
;  5509     5614  2      ! X = .DSTPTR[DST$B_LENGTH]
;  5510     5615  2      ! Y = .NEXT_DSTPTR[DST$B_LENGTH]                                                    ;
;  5511     5616  2      !
;  5512     5617  2      !   +---------------+       ^       <--- "DSTPTR" in code below
;  5513     5618  2      !   | count = X     |       |
;  5514     5619  2      !   +---------------+   "NEW_LENGTH" in code below
;  5515     5620  2      !   |  X bytes      |       |
;  5516     5621  2      !   |    ...        |       |
;  5517     5622  2      !   +---------------+       v       <-- "NEXT_DSTPTR" in code below
;  5518     5623  2      !   | count = Y     |
;  5519     5624  2      !   +---------------+
;  5520     5625  2      !   | CONTIN code   |
;  5521     5626  2      !   +---------------+       ^
;  5522     5627  2      !   |  Y-1 bytes    |   "NEXT_LEN" in code below
;  5523     5628  2      !   |    ...        |       |
;  5524     5629  2      !   +---------------+       v
;  5525     5630  2      !
;  5526     5631  2      ! Here is our situation after one or more times around the loop:
;  5527     5632  2      !
;  5528     5633  2      ! This is the DST in temporary memory that we've built so far:
;  5529     5634  2      !
;  5530     5635  2      !   +---------------+       ^       <--- "DSTPTR" in code below
;  5531     5636  2      !   | count byte    |       |
;  5532     5637  2      !   +---------------+   "NEW_LENGTH" in code below
;  5533     5638  2      !   |  X bytes      |       |
;  5534     5639  2      !   |    ...        |       |
;  5535     5640  2      !   +---------------+       v
;  5536     5641  2      !
;  5537     5642  2      ! This is where we are in the original DST:
;  5538     5643  2      !
;  5539     5644  2      !   +---------------+   <-- "NEXT_DSTPTR" in code below
;  5540     5645  2      !   | count = Y     |
;  5541     5646  2      !   +---------------+
;  5542     5647  2      !   | CONTIN code   |
;  5543     5648  2      !   +---------------+       ^
;  5544     5649  2      !   |  Y-1 bytes    |   "NEXT_LEN" in code below
;  5545     5650  2      !   |    ...        |       |
;  5546     5651  2      !   +---------------+       v
;  5547     5652  2      !
;  5548     5653  2      ! "NEW_LENGTH" always represents the length in bytes (including the
;  5549     5654  2      ! count byte) of the DST record we have built so far out of temporary
;  5550     5655  2      ! memory. This implies that NEW_LENGTH is X+1 at the start of the loop.
;  5551     5656  2      ! "NEXT_LEN" always represents the amount of bytes we are going to
;  5552     5657  2      ! append on to the DST record we are building. This is always Y-1 bytes.
;  5553     5658  2      ! NEW_LENGTH is thus incremented by NEXT_LEN each time around the loop.
;  5554     5659  2      !
;  5555     5660  2      DSTPTR = .PASSED_DSTPTR[0];
;  5556     5661  2      NEW_LENGTH = .DSTPTR[DST$B_LENGTH] + 1;
```

```
;  5557          5662  2     NEXT_DSTPTR = DSTPTR[DST$A_NEXT] + .DSTPTR[DST$B_LENGTH];
;  5558          5663  2     WHILE .NEXT_DSTPTR[DST$B_TYPE] EQL DST$K_CONTIN DO
;  5559          5664  3         BEGIN
;  5560          5665  3
;  5561          5666  3
;  5562          5667  3         ! Compute the new size and allocate enough memory for a merged record.
;  5563          5668  3         !
;  5564          5669  3         NEXT_LEN = .NEXT_DSTPTR[DST$B_LENGTH] - 1;
;  5565          5670  3         NEW_LENGTH = .NEW_LENGTH + .NEXT_LEN;
;  5566          5671  3         NEW_DSTPTR = DBG$GET_TEMPMEM((.NEW_LENGTH + 3)/4);
;  5567          5672  3
;  5568          5673  3
;  5569          5674  3         ! Move the bytes into the merged record.
;  5570          5675  3         !
;  5571          5676  3         TEMP_PTR = CH$MOVE(.NEW_LENGTH - .NEXT_LEN, .DSTPTR, .NEW_DSTPTR);
;  5572          5677  3         CH$MOVE(.NEXT_LEN, .NEXT_DSTPTR + 2, .TEMP_PTR);
;  5573          5678  3
;  5574          5679  3
;  5575          5680  3         ! Move the new DST pointer into place.
;  5576          5681  3         !
;  5577          5682  3         DSTPTR = .NEW_DSTPTR;
;  5578          5683  3
;  5579          5684  3
;  5580          5685  3         ! Set the continued flag.
;  5581          5686  3         !
;  5582          5687  3         CONTINUATION_FLAG = TRUE;
;  5583          5688  3
;  5584          5689  3
;  5585          5690  3         ! Try another.
;  5586          5691  3         !
;  5587          5692  3         NEXT_DSTPTR = NEXT_DSTPTR[DST$A_NEXT] + .NEXT_DSTPTR[DST$B_LENGTH];
;  5588          5693  2         END;
;  5589          5694  2
;  5590          5695  2
;  5591          5696  2     ! When there was a continuation record
;  5592          5697  2     !     1       We move the record to permanent memory and
;  5593          5698  2     !             release the temporary memory
;  5594          5699  2     !     2       Add an entry to the moved DST list
;  5595          5700  2     !
;  5596          5701  3     IF (.CONTINUATION_FLAG)
;  5597          5702  2     THEN
;  5598          5703  3         BEGIN
;  5599          5704  4         IF ((.DSTPTR[DST$B_TYPE] EQL DST$K_VARVAL)       OR
;  5600          5705  4             (.DSTPTR[DST$B_TYPE] EQL DST$K_BOOL)         OR
;  5601          5706  4             (.DSTPTR[DST$B_TYPE] EQL DST$K_SEPTYP)       OR
;  5602          5707  4             (.DSTPTR[DST$B_TYPE] EQL DST$K_ENUMELT)      OR
;  5603          5708  4             (.DSTPTR[DST$B_TYPE] EQL DST$K_VARBEG)       OR
;  5604          5709  4             (.DSTPTR[DST$B_TYPE] EQL DST$K_RECBEG)       OR
;  5605          5710  4             (.DSTPTR[DST$B_TYPE] EQL DST$K_TYPSPEC)      OR
;  5606          5711  4             (.DSTPTR[DST$B_TYPE] EQL DST$K_LBLORLIT)     OR
;  5607          5712  4             (.DSTPTR[DST$B_TYPE] EQL DST$K_LABEL)        OR
;  5608          5713  4             (.DSTPTR[DST$B_TYPE] EQL DST$K_OVERLOAD))
;  5609          5714  3         THEN
;  5610          5715  4             BEGIN
;  5611          5716  4
;  5612          5717  4
;  5613          5718  4             ! Move the record to permanent memory and release the memory.
```

```
;  5614          5719  4              !
;  5615          5720  4              TEMP_PTR = DBG$GET_MEMORY( (.NEW_LENGTH + 3)/4 );
;  5616          5721  4              CH$MOVE(.NEW_LENGTH, .DSTPTR, .TEMP_PTR);
;  5617          5722  4              DSTPTR = .TEMP_PTR;
;  5618          5723  4
;  5619          5724  4
;  5620          5725  4              ! Get and load the new Moved DST Entry.
;  5621          5726  4              !
;  5622          5727  4              MOVED_DST_ENTRY = DBG$GET_MEMORY(MDE$K_LENGTH);
;  5623          5728  4              MOVED_DST_ENTRY[ MDE$L_ORIGINAL_DSTPTR ] = .PASSED_DSTPTR[0];
;  5624          5729  4              MOVED_DST_ENTRY[ MDE$L_REAL_DSTPTR     ] = .DSTPTR;                    ;
;  5625          5730  4              MOVED_DST_ENTRY[ MDE$L_NEXT_DSTPTR     ] = .NEXT_DSTPTR;
;  5626          5731  4              MOVED_DST_ENTRY[ MDE$L_REAL_LENGTH     ] = .NEW_LENGTH;
;  5627          5732  4
;  5628          5733  4
;  5629          5734  4              ! Link it up.
;  5630          5735  4              !
;  5631          5736  4              MOVED_DST_ENTRY[MDE$L_FLINK] = .DBG$GL_MOVED_DST_LIST_HEAD;
;  5632          5737  4              DBG$GL_MOVED_DST_LIST_HEAD = .MOVED_DST_ENTRY;
;  5633          5738  4              END
;  5634          5739  4
;  5635          5740  3          ELSE
;  5636          5741  3              $DBG_ERROR('RSTCNTRL\MERGE_CONTINUATION_DST');
;  5637          5742  3
;  5638          5743  2          END;
;  5639          5744  2
;  5640          5745  2
;  5641          5746  2      ! Return the values.
;  5642          5747  2      !
;  5643          5748  2      PASSED_DSTPTR[0] = .DSTPTR;
;  5644          5749  2      PASSED_NEXT_DSTPTR[0] = .NEXT_DSTPTR;
;  5645          5750  2      RETURN .CONTINUATION_FLAG;
;  5646          5751  2
;  5647          5752  1      END;


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

45 47 52 45 4D 5C 4C 52 54 4E 43 54 53 52 1F  00159 P.AAU:    .ASCII  <31>\RSTCNTRL\<92>\MERGE_CONTINUATION_\     ;
   5F 4E 4F 49 54 41 55 4E 49 54 4E 4F 43 5F  00168                                                          ;
                              54 53 44  00176              .ASCII  \DST\                                      ;


                                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                        0FFC 00000 MERGE_CONTINUATION_DST:
                                                                .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 5548
                                     5E      08  C2 00002        SUBL2   #8, SP                                     ;
                                         04  AE  D4 00005        CLRL    CONTINUATION_FLAG                          ; 5591
                                     59  04  BC  D0 00008        MOVL    @PASSED_DSTPTR, DSTPTR                     ; 5660
                                     50      69  9A 0000C        MOVZBL  (DSTPTR), R0                               ; 5661
                                     56  01  A0  9E 0000F        MOVAB   1(R0), NEW_LENGTH                          ;
                                     58  01 A049 9E 00013        MOVAB   1(R0)[DSTPTR], NEXT_DSTPTR                 ; 5662
                                 AD  8F  01  A8  91 00018 1$:    CMPB    1(NEXT_DSTPTR), #173                       ; 5663
                                             3B  12 0001D        BNEQ    2$                                         ;
```

```
                        5A      68  9A 0001F        MOVZBL  (NEXT_DSTPTR), R10                              ; 5669
                        57  FF  AA  9E 00022        MOVAB   -1(R10), NEXT_LEN
                        56      57  C0 00026        ADDL2   NEXT_LEN, NEW_LENGTH                           ; 5670
                        50  03  A6  9E 00029        MOVAB   3(R6), R0                                      ; 5671
                7E      50      04  C7 0002D        DIVL3   #4, R0, -(SP)
                  00000000G     00  01  FB 00031    CALLS   #1, DBG$GET_TEMPMEM
                        6E      50  D0 00038        MOVL    R0, NEW_DSTPTR
                50              56  C3 0003B        SUBL3   NEXT_LEN, NEW_LENGTH, R0                       ; 5676
             00 BE              69  28 0003F        MOVC3   R0, (DSTPTR), @NEW_DSTPTR
                        5B      53  D0 00044        MOVL    R3, TEMP_PTR
                6B      02      57  28 00047        MOVC3   NEXT_LEN, 2(NEXT_DSTPTR), (TEMP_PTR)           ; 5677
                        59      6E  D0 0004C        MOVL    NEW_DSTPTR, DSTPTR                             ; 5682
                        04      01  D0 0004F        MOVL    #1, CONTINUATION_FLAG                          ; 5687
                        58  01 AA48  9E 00053        MOVAB   1(R10)[NEXT_DSTPTR], NEXT_DSTPTR               ; 5692
                            BE      11 00058        BRB     1$                                             ; 5663
                        03  04  AE  E8 0005A 2$:     BLBS    CONTINUATION_FLAG, 3$                          ; 5701
                          0098  31  00005E          BRW     6$
                        50  01  A9  9A 00061 3$:     MOVZBL  1(DSTPTR), R0                                  ; 5704
                    9D  8F      50  91 00065        CMPB    R0, #157
                            36  13 00069        BEQL    4$
                    9E  8F      50  91 0006B        CMPB    R0, #158                                       ; 5705
                            30  13 0006F        BEQL    4$
                    A3  8F      50  91 00071        CMPB    R0, #163                                       ; 5706
                            2A  13 00075        BEQL    4$
                    A4  8F      50  91 00077        CMPB    R0, #164                                       ; 5707
                            24  13 0007B        BEQL    4$
                    A7  8F      50  91 0007D        CMPB    R0, #167                                       ; 5708
                            1E  13 00081        BEQL    4$
                    AB  8F      50  91 00083        CMPB    R0, #171                                       ; 5709
                            18  13 00087        BEQL    4$
                    AF  8F      50  91 00089        CMPB    R0, #175                                       ; 5710
                            12  13 0008D        BEQL    4$
                    BA  8F      50  91 0008F        CMPB    R0, #186                                       ; 5711
                            0C  13 00093        BEQL    4$
                    BB  8F      50  91 00095        CMPB    R0, #187                                       ; 5712
                            06  13 00099        BEQL    4$
                    A9  8F      50  91 0009B        CMPB    R0, #169                                       ; 5713
                            43  12 0009F        BNEQ    5$
                        50      03  A6  9E 000A1 4$:     MOVAB   3(R6), R0                                  ; 5720
                7E      50          04  C7 000A5        DIVL3   #4, R0, -(SP)
                  00000000G     00  01  FB 000A9    CALLS   #1, DBG$GET_MEMORY
                        5B      50  D0 000B0        MOVL    R0, TEMP_PTR
                6B      69      56  28 000B3        MOVC3   NEW_LENGTH, (DSTPTR), (TEMP_PTR)               ; 5721
                        59      5B  D0 000B7        MOVL    TEMP_PTR, DSTPTR                               ; 5722
                                05  DD 000BA        PUSHL   #5                                             ; 5727
                  00000000G     00  01  FB 000BC    CALLS   #1, DBG$GET_MEMORY
                        04  A0      BC  D0 000C3        MOVL    @PASSED_DSTPTR, 4(MOVED_DST_ENTRY)         ; 5728
                        08  A0      59  D0 000C8        MOVL    DSTPTR, 8(MOVED_DST_ENTRY)                 ; 5729
                        10  A0      58  D0 000CC        MOVL    NEXT_DSTPTR, 16(MOVED_DST_ENTRY)           ; 5730
                        0C  A0      56  D0 000D0        MOVL    NEW_LENGTH, 12(MOVED_DST_ENTRY)            ; 5731
                            60  00000000G   00  D0 000D4        MOVL    DBG$GL_MOVED_DST_LIST_HEAD, -      ; 5736
                                                                    (MOVED_DST_ENTRY)
                  00000000G     00          50  D0 000DB        MOVL    MOVED_DST_ENTRY, DBG$GL_MOVED_DST_LIST_HEAD ; 5737
                                            15  11 000E2        BRB     6$                                 ; 5704
                                00000000'   EF  9F 000E4 5$:     PUSHAB  P.AAU                              ; 5741
                                            01  DD 000EA        PUSHL   #1
                                00028362    8F  DD 000F0        PUSHL   #164706
```

```
                00000000G  00          03  FB 000F2          CALLS   #3, LIB$SIGNAL                          ;
                      04   BC          59  D0 000F9 6$:      MOVL    DSTPTR, @PASSED_DSTPTR                 ; 5748
                      08   BC          58  D0 000FD          MOVL    NEXT_DSTPTR, @PASSED_NEXT_DSTPTR       ; 5749
                           50      04   AE  D0 00101          MOVL    CONTINUATION_FLAG, R0                  ; 5750
                                           04 00105          RET                                            ; 5752

; Routine Size:  262 bytes,    Routine Base:  DBG$CODE + 29EE
```

```
;5649          5753  1 ROUTINE POSITION_GST(GST_REC_FLAG) =
;5650          5754  1 !
;5651          5755  1 ! FUNCTION
;5652          5756  1 !       This routine positions the GST at the start of the GST or at the next
;5653          5757  1 !       sequential record, depending on the input parameter.  It expects the
;5654          5758  1 !       OWN variable GST$BEGIN_ADDR to contain the address of the first GST
;5655          5759  1 !       record and it expects GST$NEXT_ADDR to point to the "current" GST
;5656          5760  1 !       record.  GST$NEXT_ADDR is updated by this routine as the "next sequen-
;5657          5761  1 !       tial record" is updated.  The routine returns the address of the record
;5658          5762  1 !       to which GST$NEXT_ADDR is then pointing as its value.
;5659          5763  1 !
;5660          5764  1 ! INPUTS
;5661          5765  1 !       GST_REC_FLAG - If this flag is non-zero, the GST is repositioned to its
;5662          5766  1 !                      first record and the address of that record is returned as
;5663          5767  1 !                      the routine value.  If this flag is zero, the address of the
;5664          5768  1 !                      next sequential GST record is returned as the routine value.
;5665          5769  1 !
;5666          5770  1 ! OUTPUTS
;5667          5771  1 !       If GST_REC_FLAG is non-zero, the address of the first record in the GST
;5668          5772  1 !                      is returned as the routine value.  If GST_REC_FLAG is zero,
;5669          5773  1 !                      the address of the next sequential GST record is returned as
;5670          5774  1 !                      the routine value.  If there is no next sequential GST record,
;5671          5775  1 !                      zero is returned as the routine value.
;5672          5776  1 !
;5673          5777  1
;5674          5778  2     BEGIN
;5675          5779  2
;5676          5780  2     OWN
;5677          5781  2         RECORDS_LEFT;                           ! Number of GST left
;5678          5782  2
;5679          5783  2     LOCAL
;5680          5784  2         BLOCK_ADDR,                             ! GST address
;5681          5785  2         GST_NEXT_ADDR: REF VECTOR[,WORD];       ! Virtual Address of next GST
;5682          5786  2
;5683          5787  2
;5684          5788  2
;5685          5789  2     ! If there is no mapped GST, signal error.
;5686          5790  2     !
;5687          5791  2     IF (.GST$BEGIN_ADDR LEQA 1) THEN SIGNAL(DBG$_INVGSTREC);
;5688          5792  2
;5689          5793  2
;5690          5794  2     ! Flag is cleared to indicate the pointer should be re-positioned to start.
;5691          5795  2     !
;5692          5796  2     IF NOT .GST_REC_FLAG
;5693          5797  2     THEN
;5694          5798  3         BEGIN
;5695          5799  3         GST$NEXT_ADDR = .GST$BEGIN_ADDR;
;5696          5800  3         RECORDS_LEFT  = .GST$REC_COUNT;
;5697          5801  2         END;
;5698          5802  2
;5699          5803  2
;5700          5804  2     ! Stop the following from faulting if some caller ignores the end condition
;5701          5805  2     ! and effectively causes us to 'run off the end' of the mapped GST.
;5702          5806  2     !
;5703          5807  2     IF (.RECORDS_LEFT LSS 1) THEN RETURN (0);
;5704          5808  2
;5705          5809  2
```

```
;  5706          5810  2     ! Pick up the address of the current record, and update the pointer to the
;  5707          5811  2     ! subsequent one.
;  5708          5812  2     !
;  5709          5813  2     BLOCK_ADDR = .GST$NEXT_ADDR + 2;
;  5710          5814  2     GST_NEXT_ADDR = .GST$NEXT_ADDR;
;  5711          5815  2     GST$NEXT_ADDR = .GST$NEXT_ADDR + 2 + ((.GST_NEXT_ADDR[0] +1)/2)*2;
;  5712          5816  2     RECORDS_LEFT = .RECORDS_LEFT - 1;
;  5713          5817  2     RETURN (.BLOCK_ADDR);
;  5714          5818  2
;  5715          5819  1     END;


                                                        .PSECT  DBG$OWN,NOEXE,  PIC,2

                                                04EEC RECORDS_LEFT:
                                                        .BLKB   4


                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                001C 00000 POSITION_GST:
                                                        .WORD   Save R2,R3,R4                           ;  5753
                                    54 00000000'  EF  9E 00002          MOVAB   RECORDS_LEFT, R4
                                    53 00000000'  EF  9E 00009          MOVAB   GST$NEXT_ADDR, R3
                                    01        FC  A3  D1 00010          CMPL    GST$BEGIN_ADDR, #1          ;  5791
                                                  0D  1A 00014          BGTRU   1$
                                        0002835A  8F  DD 00016          PUSHL   #164698
                                00000000G   00        01 FB 0001C       CALLS   #1, LIB$SIGNAL
                                    08        04  AC  E8 00023 1$:      BLBS    GST_REC_FLAG, 2$            ;  5796
                                    63        FC  A3  D0 00027          MOVL    GST$BEGIN_ADDR, GST$NEXT_ADDR ;  5799
                                    64        04  A3  D0 0002B          MOVL    GST$REC_COUNT, RECORDS_LEFT ;  5800
                                                  64  D5 0002F 2$:      TSTL    RECORDS_LEFT                ;  5807
                                                  1A  15 00031          BLEQ    3$
                                    52            63  D0 00033          MOVL    GST$NEXT_ADDR, R2           ;  5813
                                    50        02  A2  9E 00036          MOVAB   2(R2), BLOCK_ADDR
                                    51            52  D0 0003A          MOVL    R2, GST_NEXT_ADDR           ;  5814
                                    51            61  3C 0003D          MOVZWL  (GST_NEXT_ADDR), R1         ;  5815
                                                  51  D6 00040          INCL    R1
                                    51            02  C6 00042          DIVL2   #2, R1
                                    63        02 A241  3E 00045          MOVAW   2(R2)[R1], GST$NEXT_ADDR
                                                  64  D7 0004A          DECL    RECORDS_LEFT                ;  5816
                                                  04  04 0004C          RET                                 ;  5817
                                                  50  D4 0004D 3$:      CLRL    R0                          ;  5819
                                                  04  04 0004F          RET

; Routine Size:  80 bytes,    Routine Base:  DBG$CODE + 2AF4
```

```
;  5717          5820  1 ROUTINE READ_DEBUG_MODULE_TABLE(IMGFILCHAN, IMG_DST_BLOCK, EST_RST_SIZE) =
;  5718          5821  1 !
;  5719          5822  1 ! FUNCTION
;  5720          5823  1 !       This routine reads in the Debug Module Table (the DMT) that the linker
;  5721          5824  1 !       puts in the executable image and uses this table to initialize the Run-
;  5722          5825  1 !       Time Symbol Table (the RST).  This table exists in images built by the
;  5723          5826  1 !       VMS V4.0 and later linkers, and does not exist in images built by ear-
;  5724          5827  1 !       lier linkers.  The advantage of this table is that it allows us to
;  5725          5828  1 !       initialize the RST very rapidly compared to making a complete scan of
;  5726          5829  1 !       the Debug Symbol Table (the DST), which must otherwise be done.
;  5727          5830  1 !
;  5728          5831  1 !       The initialization takes place as follows: First the DMT is mapped into
;  5729          5832  1 !       virtual memory.  Then the whole DMT is scanned, and for each DMT entry,
;  5730          5833  1 !       a Module RST Entry is built.  The Module's DST pointer is retrieved
;  5731          5834  1 !       directly from the DMT entry and the module size (i.e., the amount of
;  5732          5835  1 !       Run-Time Symbol Table space it would require if SET) is computed by
;  5733          5836  1 !       taking the size of the module's DST (as given by the DMT entry) and
;  5734          5837  1 !       multiplying it by a scale factor which yields a rough approximation to
;  5735          5838  1 !       the actual RST size.  The estimated RST sizes of the MIN_MODS_IN_RST
;  5736          5839  1 !       largest modules are remembered in a table as the DMT scan proceeds.
;  5737          5840  1 !
;  5738          5841  1 !       The module language code is not set in the Module RST Entry at this
;  5739          5842  1 !       time.  This means that no access to the DST is necessary during initia-
;  5740          5843  1 !       lization which prevents any DST paging.  The initialization is thus
;  5741          5844  1 !       speeded up.  The language code is retrieved when the module is SET.
;  5742          5845  1 !
;  5743          5846  1 !       For each module represented in the DMT (and thus in the DST), the DMT
;  5744          5847  1 !       lists the addresses and lengths of all its PSECTs.  This information
;  5745          5848  1 !       is used to build the Program Static Address Table (SAT) during the DMT
;  5746          5849  1 !       scan.
;  5747          5850  1 !
;  5748          5851  1 !       At the end of the scan, the estimated total RST memory requirements of
;  5749          5852  1 !       the several largest modules is computed and returned to the caller.
;  5750          5853  1 !       This number is used by DBG$RST_INIT to determine the initial size of
;  5751          5854  1 !       the DEBUG memory pool.
;  5752          5855  1 !
;  5753          5856  1 ! INPUTS
;  5754          5857  1 !       IMGFILCHAN - The executable image file channel number.  The image is
;  5755          5858  1 !                    already open on this channel; the DMT can thus be read on it.
;  5756          5859  1 !
;  5757          5860  1 !       IMG_DST_BLOCK - The address of the Image Header Symbol Information
;  5758          5861  1 !                    block which gives the locations (virtual blocks numbers) and
;  5759          5862  1 !                    sizes of the DST, the GST, and the DMT.  Here we use this
;  5760          5863  1 !                    block to get the DMT location and size.
;  5761          5864  1 !
;  5762          5865  1 !       EST_RST_SIZE - The address of a longword to receive the estimated number
;  5763          5866  1 !                    of longwords of symbol table space needed to hold the RST of
;  5764          5867  1 !                    the MIN_MODS_IN_RST largest modules.
;  5765          5868  1 !
;  5766          5869  1 ! OUTPUTS
;  5767          5870  1 !       EST_RST_SIZE - The estimated number of longwords needed to hold the RST
;  5768          5871  1 !                    for the MIN_MODS_IN_RST largest modules in the image is re-
;  5769          5872  1 !                    turned to the EST_RST_SIZE location.  This number is estimated
;  5770          5873  1 !                    by multiplying the DST size of each module by a scale factor
;  5771          5874  1 !                    which roughly estimates the corresponding RST size.
;  5772          5875  1 !
;  5773          5876  1 !       If the RST was successfully initialized using the Debug Module Table,
```

```
;  5774          5877  1 !                TRUE is returned as the routine value.  If the initialization
;  5775          5878  1 !                failed for any reason, FALSE is returned so that the initial-
;  5776          5879  1 !                ization can be tried the slow way by reading the DST.
;  5777          5880  1 !
;  5778          5881  1
;  5779          5882  2     BEGIN
;  5780          5883  2
;  5781          5884  2     MAP
;  5782          5885  2         IMG_DST_BLOCK: REF IHS$ENTRY,    ! Pointer to the DST/GST/DMT descriptor
;  5783          5886  2                                          !      in the executable image header
;  5784          5887  2         EST_RST_SIZE: REF VECTOR[1];     ! Estimated RST size parameter
;  5785          5888  2
;  5786          5889  2     LITERAL
;  5787          5890  2         DST_TO_RST_FACTOR = 25;          ! Scale factor (times 10) for convert-
;  5788          5891  2                                          !      ing DST size to estimated RST
;  5789          5892  2                                          !      size for current module
;  5790          5893  2
;  5791          5894  2     LOCAL
;  5792          5895  2         DMTPTR: REF VECTOR[,LONG],       ! Pointer to current Debug Module Table
;  5793          5896  2                                          !      location
;  5794          5897  2         DMT_SIZE,                        ! The total size of the DMT in bytes
;  5795          5898  2         DMT_START_ADDR,                  ! The start address of the DMT
;  5796          5899  2         DSTPTR,                          ! Pointer to Module Begin DST Record
;  5797          5900  2         EXE_SECBOUNDS: VECTOR[2,LONG],   ! Map-section bounds descriptor for
;  5798          5901  2                                          !      SYS$CRMPSC system service
;  5799          5902  2         EXE_SECNAM_DESC: VECTOR[2,LONG],! Map-section name descriptor for the
;  5800          5903  2                                          !      SYS$CRMPSC system service
;  5801          5904  2         MOD_SIZE_TBL:                    ! Table giving the estimated RST sizes
;  5802          5905  2                 VECTOR[MIN_MODS_IN_RST],!      of the MIN_MODS_IN_RST largest
;  5803          5906  2                                          !      modules in longwords.
;  5804          5907  2         NPSECTS,                         ! Number of PSECTs in current module
;  5805          5908  2         RSTPTR: REF RST$ENTRY,           ! Pointer to current Module RST Entry
;  5806          5909  2         RSTSIZE,                         ! The estimated RST size in longwords
;  5807          5910  2                                          !      of current module's symbol table
;  5808          5911  2         STATUS: BLOCK[1, LONG];          ! The status returned by SYS$CRMPSC
;  5809          5912  2
;  5810          5913  2
;  5811          5914  2
;  5812          5915  2     ! Check that there is a Debug Module Table (DMT) in this image.  If the
;  5813          5916  2     ! DMT virtual block number or length is zero, there is none and we return
;  5814          5917  2     ! FALSE to the caller.  This will happen if the user did not LINK/DEBUG.
;  5815          5918  2     !
;  5816          5919  2     EST_RST_SIZE[0] = 0;
;  5817          5920  2     IF (.IMG_DST_BLOCK[IHS$L_DMTVBN] EQL 0) OR
;  5818          5921  3        (.IMG_DST_BLOCK[IHS$L_DMTBYTES] EQL 0)
;  5819          5922  2     THEN
;  5820          5923  2         RETURN FALSE;
;  5821          5924  2
;  5822          5925  2
;  5823          5926  2     ! Map the Debug Module Table (the DMT) into our virtual address space.
;  5824          5927  2     ! After doing so, we set DMT_START_ADDR to point to this table.  We also
;  5825          5928  2     ! set DMT_SIZE to contain its size in bytes.  If we are unable to map it
;  5826          5929  2     ! in, we signal the bad status and return FALSE so that DBG$RST_INIT can
;  5827          5930  2     ! initialize the RST the slow way by reading through the whole DST.
;  5828          5931  2     !
;  5829          5932  2     DMT_SIZE = .IMG_DST_BLOCK[IHS$L_DMTBYTES];
;  5830          5933  2     EXE_SECBOUNDS[0] = 200;
```

```
;  5831          5934  2      EXE_SECBOUNDS[1] = 1^30;
;  5832          5935  2      EXE_SECNAM_DESC[0] = 3;
;  5833          5936  2      EXE_SECNAM_DESC[1] = UPLIT BYTE(%ASCIC 'DMT');
;  5834          5937  2      STATUS = SYS$CRMPSC(EXE_SECBOUNDS, EXE_SECBOUNDS, 0, SEC$M_EXPREG,
;  5835          5938  2                          EXE_SECNAM_DESC, 0, 0, .IMGFILCHAN,
;  5836          5939  2                          (.DMT_SIZE + 511)/512,
;  5837          5940  2                          .IMG_DST_BLOCK[IHS$L_DMTVBN], 0, 0);
;  5838          5941  2      IF NOT .STATUS
;  5839          5942  2      THEN
;  5840          5943  3          BEGIN
;  5841          5944  3          STATUS[STS$V_SEVERITY] = SYS$K_INFO;
;  5842          5945  3          SIGNAL(.STATUS);
;  5843          5946  3          RETURN FALSE;
;  5844          5947  2          END;
;  5845          5948  2
;  5846          5949  2      DMT_START_ADDR = .EXE_SECBOUNDS[0];
;  5847          5950  2
;  5848          5951  2
;  5849          5952  2      ! Set up a table to keep track of the MIN_MODS_IN_RST largest modules in
;  5850          5953  2      ! the DST.  Also initialize PREV_RST_PTR.
;  5851          5954  2      !
;  5852          5955  2      INCR I FROM 0 TO MIN_MODS_IN_RST - 1 DO
;  5853          5956  2          MOD_SIZE_TBL[.I] = 0;
;  5854          5957  2
;  5855          5958  2      PREV_RST_PTR = .RST$START_ADDR;
;  5856          5959  2
;  5857          5960  2
;  5858          5961  2      ! Now loop through the Debug Module Table (the DMT) to get each module's
;  5859          5962  2      ! Module Begin DST Record address, size, and static address ranges.
;  5860          5963  2      !
;  5861          5964  2      DMTPTR = .DMT_START_ADDR;
;  5862          5965  2      WHILE .DMTPTR LSSA .DMT_START_ADDR + .DMT_SIZE DO
;  5863          5966  3          BEGIN
;  5864          5967  3
;  5865          5968  3
;  5866          5969  3          ! Get the DST pointer from the current DMT entry and check that it
;  5867          5970  3          ! actually points to a Module Begin DST Record.  Note that if we go
;  5868          5971  3          ! past DST$END_ADDR and this is a Testable Debugger, we have reached
;  5869          5972  3          ! the DEBUG part of the DST so we simply exit the loop, having already
;  5870          5973  3          ! initialized the RST for the test program part of the image.
;  5871          5974  3          !
;  5872          5975  3          DSTPTR = .DST$BEGIN_ADDR + .DMTPTR[0];
;  5873          5976  3          IF .DSTPTR LSSA .DST$BEGIN_ADDR
;  5874          5977  3          THEN
;  5875          5978  4              BEGIN
;  5876          5979  4              SIGNAL(DBG$_INVDMTPTR);
;  5877          5980  4              RETURN FALSE;
;  5878          5981  4              END;
;  5879          5982  3
;  5880          5983  3          IF .DSTPTR GEQA .DST$END_ADDR
;  5881          5984  3          THEN
;  5882          5985  4              BEGIN
;  5883          5986  4              IF .DBG$GV_CONTROL[DBG$V_CONTROL_TDBG] THEN EXITLOOP;
;  5884          5987  4              SIGNAL(DBG$_INVDMTPTR);
;  5885          5988  4              RETURN FALSE;
;  5886          5989  3              END;
;  5887          5990  3
```

```
;  5888          5991  3
;  5889          5992  3          ! Build the Module RST Entry for the current module in the DMT.  Note
;  5890          5993  3          ! that the language code is not set: this saves us an access to the DST
;  5891          5994  3          ! and so prevents any DST paging during the RST initialization.  Also
;  5892          5995  3          ! link that RST entry into the RST Module Chain.
;  5893          5996  3          !
;  5894          5997  3          RSTPTR = GET_MEM(RST$K_MODENTSIZ);
;  5895          5998  3          RSTPTR[RST$L_DSTPTR] = .DSTPTR;
;  5896          5999  3          RSTPTR[RST$B_KIND] = RST$K_MODULE;
;  5897          6000  3          RSTPTR[RST$B_LANGUAGE] = DBG$K_UNKNOWN;
;  5898          6001  3          PREV_RST_PTR[RST$L_NXTMODPTR] = .RSTPTR;
;  5899          6002  3          PREV_RST_PTR = .RSTPTR;
;  5900          6003  3
;  5901          6004  3
;  5902          6005  3          ! Compute the estimated RST size of this module from the actual DST size
;  5903          6006  3          ! as given by the DMT entry.  Store this size (in bytes) in the Module
;  5904          6007  3          ! RST Entry.  Then, if this RST size makes this one of the several larg-
;  5905          6008  3          ! est modules, enter it in MOD_SIZE_TBL.  Note that this table remains
;  5906          6009  3          ! sorted on module size (given in longwords), largest modules first.
;  5907          6010  3          !
;  5908          6011  3          RSTSIZE = (DST_TO_RST_FACTOR*.DMTPTR[1])/(10*%UPVAL);
;  5909          6012  3          RSTPTR[RST$L_MODRSTSIZ] = .RSTSIZE*%UPVAL;
;  5910          6013  3          INCR I FROM 0 TO MIN_MODS_IN_RST - 1 DO
;  5911          6014  4              BEGIN
;  5912          6015  4              IF .RSTSIZE GTR .MOD_SIZE_TBL[.I]
;  5913          6016  4              THEN
;  5914          6017  5                  BEGIN
;  5915          6018  5                  DECR J FROM MIN_MODS_IN_RST - 2 TO .I DO
;  5916          6019  5                      MOD_SIZE_TBL[.J+1] = .MOD_SIZE_TBL[.J];
;  5917          6020  5
;  5918          6021  5                  MOD_SIZE_TBL[.I] = .RSTSIZE;
;  5919          6022  5                  EXITLOOP;
;  5920          6023  4                  END;
;  5921          6024  4
;  5922          6025  3              END;
;  5923          6026  3
;  5924          6027  3
;  5925          6028  3          ! Now loop through the PSECT entries for this module to build the
;  5926          6029  3          ! Program Static Address Table (SAT) entries for this module.
;  5927          6030  3          !
;  5928          6031  3          NPSECTS = .DMTPTR[2];
;  5929          6032  3          DMTPTR = DMTPTR[3];
;  5930          6033  3          INCR I FROM 1 TO .NPSECTS DO
;  5931          6034  4              BEGIN
;  5932          6035  4              IF .DMTPTR[1] GTR 0
;  5933          6036  4              THEN
;  5934          6037  4                  STATIC_ADDRESS(.DMTPTR[0], .DMTPTR[1],
;  5935          6038  4                                 .RSTPTR, SAT$START_ADDR, FALSE);
;  5936          6039  4              DMTPTR = DMTPTR[2];
;  5937          6040  4              END;
;  5938          6041  3
;  5939          6042  2          END;                            ! End of loop over the DMT
;  5940          6043  2
;  5941          6044  2
;  5942          6045  2      ! We have now read through the entire Debug Module Table and built the
;  5943          6046  2      ! corresponding RST Module Chain.  Return the estimated RST size (in long-
;  5944          6047  2      ! words) for the MIN_MODS_IN_RST largest modules.
```

```
;  5945          6048  2     !
;  5946          6049  2     RSTSIZE = 0;
;  5947          6050  2     INCR I FROM 0 TO MIN_MODS_IN_RST - 1 DO
;  5948          6051  2         RSTSIZE = .RSTSIZE + .MOD_SIZE_TBL[.I];
;  5949          6052  2
;  5950          6053  2     EST_RST_SIZE[0] = .RSTSIZE;
;  5951          6054  2
;  5952          6055  2
;  5953          6056  2     ! Finally return successfully to the caller.
;  5954          6057  2     !
;  5955          6058  2     RETURN TRUE;
;  5956          6059  2
;  5957          6060  1     END;


                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                54  4D  44  03  00179 P.AAV:  .ASCII  <3>\DMT\                                    ;


                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                    07FC 00000 READ_DEBUG_MODULE_TABLE:
                                                .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10                    ; 5820
                            5A 00000000'  EF  9E 00002  MOVAB   PREV_RST_PTR, R10                           ;
                            59 00000000'  EF  9E 00009  MOVAB   DST$BEGIN_ADDR, R9                          ;
                            5E            28  C2 00010  SUBL2   #40, SP                                     ;
                                     0C   BC  D4 00013  CLRL    @EST_RST_SIZE                               ; 5919
                            50       08   AC  D0 00016  MOVL    IMG_DST_BLOCK, R0                           ; 5920
                                     0C   A0  D5 0001A  TSTL    12(R0)                                      ;
                                          03  13 0001D  BEQL    1$                                          ;
                                     10   A0  D5 0001F  TSTL    16(R0)                                      ; 5921
                                          03  12 00022  1$:     BNEQ    2$                                  ;
                                        012D  31 00024  BRW     21$                                         ;
                                53        10   A0  D0 00027  2$:     MOVL    16(R0), DMT_SIZE            ; 5932
                             20 AE        C8   8F  9A 0002B  MOVZBL  #200, EXE_SECBOUNDS                   ; 5933
                             24 AE 40000000    8F  D0 00030  MOVL    #1073741824, EXE_SECBOUNDS+4          ; 5934
                             18 AE            03  D0 00038  MOVL    #3, EXE_SECNAM_DESC                   ; 5935
                             1C AE 00000000'  EF  9E 0003C  MOVAB   P.AAV, EXE_SECNAM_DESC+4              ; 5936
                                              7E  7C 00044  CLRQ    -(SP)                                 ; 5937
                                     0C       A0  DD 00046  PUSHL   12(R0)                                ; 5940
                                50       01FF  C3  9E 00049  MOVAB   511(R3), R0                          ; 5939
                         7E             50 00000200  8F  C7 0004E  DIVL3   #512, R0, -(SP)                ;
                                        04   AC  DD 00056  PUSHL   IMGFILCHAN                             ; 5938
                                              7E  7C 00059  CLRQ    -(SP)                                 ; 5937
                                        34   AE  9F 0005B  PUSHAB  EXE_SECNAM_DESC                        ;
                                    00020000  8F  DD 0005E  PUSHL   #131072                               ;
                                              7E  D4 00064  CLRL    -(SP)                                 ;
                                        48   AE  9F 00066  PUSHAB  EXE_SECBOUNDS                          ;
                                        4C   AE  9F 00069  PUSHAB  EXE_SECBOUNDS                          ;
                    00000000G           00        0C  FB 0006C  CALLS   #12, SYS$CRMPSC                   ;
                                        09        50  E8 00073  BLBS    STATUS, 3$                          ; 5941
                         50         03  00        03  F0 00076  INSV    #3, #0, #3, STATUS                  ; 5944
                                                  50  DD 0007B  PUSHL   STATUS                            ; 5945
                                                  3C  11 0007D  BRB     9$                                  ;
```

```
                        50        20   AE   D0 0007F 3$:     MOVL    EXE_SECBOUNDS, DMT_START_ADDR                ; 5949
                                       51   D4 00083         CLRL    I                                            ; 5955
                                   6E41   D4 00085 4$:      CLRL    MOD_SIZE_TBL[I]                              ; 5956
                     F9                51   F3 00088         AOBLEQ  #5, I, 4$                                    ;
                                       6A   A9 0008C         MOVL    RST$START_ADDR, PREV_RST_PTR                 ; 5958
                                       52   D0 00090         MOVL    DMT_START_ADDR, DMTPTR                       ; 5964
                     57                50   C1 00093         ADDL3   DMT_SIZE, DMT_START_ADDR, R7                 ; 5965
                                       57   D1 00097 5$:      CMPL    DMTPTR, R7                                   ;
                                            1F 0009A         BLSSU   7$                                           ;
                                       31   31 0009C 6$:      BRW     19$                                          ;
                     56                69   C1 0009F 7$:     ADDL3   (DMTPTR), DST$BEGIN_ADDR, DSTPTR             ; 5975
                                       69   D1 000A3         CMPL    DSTPTR, DST$BEGIN_ADDR                       ; 5976
                                            1F 000A6         BLSSU   8$                                           ;
                           04          A9   D1 000A8         CMPL    DSTPTR, DST$END_ADDR                         ; 5983
                                            1F 000AC         BLSSU   10$                                          ;
                                 E7 00000000G   E8 000AE     BLBS    DBG$GV_CONTROL, 6$                           ; 5986
                                    000286EB    DD 000B5 8$:      PUSHL   #165611                                 ; 5987
                00000000G           01   FB 000BB 9$:      CALLS   #1, LIB$SIGNAL                               ;
                                       8F   31 000C2         BRW     21$                                          ; 5988
                                       0C   DD 000C5 10$:     PUSHL   #12                                          ; 5997
                         FC93          CF   FB 000C7         CALLS   #1, GET_MEM                                  ;
                                       53   D0 000CC         MOVL    R0, RSTPTR                                   ;
                           0C          A3   D0 000CF         MOVL    DSTPTR, 12(RSTPTR)                           ; 5998
                           14          A3   01 90 000D3      MOVB    #1, 20(RSTPTR)                               ; 5999
                           29          A3   0A 90 000D7      MOVB    #10, 41(RSTPTR)                              ; 6000
                                       50   6A D0 000DB      MOVL    PREV_RST_PTR, R0                             ; 6001
                           10          A0   53 D0 000DE      MOVL    RSTPTR, T6(R0)                               ;
                                       6A   53 D0 000E2      MOVL    RSTPTR, PREV_RST_PTR                         ; 6002
                     50    04          A2   19 C5 000E5      MULL3   #25, 4(DMTPTR), R0                           ; 6011
                     55                50   28 C7 000EA      DIVL3   #40, R0, RSTSIZE                             ;
                20   A3                55   02 78 000EE      ASHL    #2, RSTSIZE, 32(RSTPTR)                      ; 6012
                                            51 D4 000F3      CLRL    I                                            ; 6013
                                   6E41     55 D1 000F5 11$:     CMPL    RSTSIZE, MOD_SIZE_TBL[I]                 ; 6015
                                            18 15 000F9      BLEQ    14$                                          ;
                                       50   04 D0 000FB      MOVL    #4, J                                        ; 6018
                                            08 11 000FE      BRB     13$                                          ;
                              04 AE40       6E40 D0 00100 12$:     MOVL    MOD_SIZE_TBL[J], MOD_SIZE_TBL+4[J]       ; 6019
                                            50 D7 00106      DECL    J                                            ;
                                       51   50 D1 00108 13$:     CMPL    J, I                                     ;
                                            F3 18 0010B      BGEQ    12$                                          ;
                                   6E41     55 D0 0010D      MOVL    RSTSIZE, MOD_SIZE_TBL[I]                     ; 6021
                                            04 11 00111      BRB     15$                                          ; 6017
                     DE                51   05 F3 00113 14$:     AOBLEQ  #5, I, 11$                               ; 6013
                                       58   A2 D0 00117 15$:     MOVL    8(DMTPTR), NPSECTS                       ; 6031
                                       52   0C C0 0011B      ADDL2   #12, DMTPTR                                  ; 6032
                                            54 D4 0011E      CLRL    I                                            ; 6033
                                            17 11 00120      BRB     18$                                          ;
                                       04   A2 D5 00122 16$:     TSTL    4(DMTPTR)                                ; 6035
                                            0F 15 00125      BLEQ    17$                                          ;
                                            7E D4 00127      CLRL    -(SP)                                        ; 6037
                                       30   A9 9F 00129      PUSHAB  SAT$START_ADDR                               ;
                                            53 DD 0012C      PUSHL   RSTPTR                                       ; 6038
                                       7E   62 7D 0012E      MOVQ    (DMTPTR), -(SP)                              ; 6037
                           0000V       CF   05 FB 00131      CALLS   #5, STATIC_ADDRESS                           ;
                                       52   08 C0 00136 17$:     ADDL2   #8, DMTPTR                               ; 6039
                     E5                54   58 F3 00139 18$:     AOBLEQ  NPSECTS, I, 16$                          ; 6033
                                      FF57  31 0013D         BRW     5$                                           ; 5965
```

```
                                55  D4 00140 19$:   CLRL    RSTSIZE                          ; 6049
                                50  D4 00142        CLRL    I                                ; 6050
                           55       6E40  C0 00144 20$:   ADDL2   MOD_SIZE_TBL[I], RSTSIZE   ; 6051
                      F8   50        05  F3 00148        AOBLEQ  #5, I, 20$                 ;
                 0C   BC   55        55  D0 0014C        MOVL    RSTSIZE, @EST_RST_SIZE     ; 6053
                           50        01  D0 00150        MOVL    #1, R0                     ; 6058
                                         04 00153        RET                                ;
                                50  D4 00154 21$:   CLRL    R0                               ; 6060
                                         04 00156        RET                                ;

; Routine Size:  343 bytes,    Routine Base:  DBG$CODE + 2B44
```

```
;  5959          6061  1 ROUTINE RST_BUILD_ERROR_HANDLER(SIGARG, MECHARG, ENBLARG) =
;  5960          6062  1 !
;  5961          6063  1 ! FUNCTION
;  5962          6064  1 !       This routine is the error handler for DBG$RST_BUILD.  The only error
;  5963          6065  1 !       conditions it checks for are DBG$_NOFREE, i.e. running out of memory
;  5964          6066  1 !       while building the RST for some module, and DBG$_ABORTED, i.e. the user
;  5965          6067  1 !       entered Control-Y DEBUG to abort the current command.  When either of
;  5966          6068  1 !       these conditions occurs, this handler calls DBG$RST_REMOVE to release
;  5967          6069  1 !       all the RST space which was allocated during the attempt to build the
;  5968          6070  1 !       module's RST.  It then resignals the error.
;  5969          6071  1 !
;  5970          6072  1 ! INPUTS
;  5971          6073  1 !       SIGARG  - The address of the signal argument vector.
;  5972          6074  1 !
;  5973          6075  1 !       MECHARG - The address of the mechanism argument vector.
;  5974          6076  1 !
;  5975          6077  1 !       ENBLARG - The address of the enable argument vector.
;  5976          6078  1 !
;  5977          6079  1 ! OUTPUTS
;  5978          6080  1 !       The error is always resignalled, meaning that SS$_RESIGNAL is returned
;  5979          6081  1 !               as the routine value.
;  5980          6082  1 !
;  5981          6083  1
;  5982          6084  2     BEGIN
;  5983          6085  2
;  5984          6086  2     MAP
;  5985          6087  2         SIGARG: REF VECTOR[,LONG];      ! The signal argument vector
;  5986          6088  2
;  5987          6089  2     LOCAL
;  5988          6090  2         RSTPTR: REF RST$ENTRY;          ! Pointer to the Module RST Entry
;  5989          6091  2
;  5990          6092  2
;  5991          6093  2     ! If we ran out of memory or were aborted by Control-Y DEBUG, remove the
;  5992          6094  2     ! partially built module from the RST.  Also, mark the module as being
;  5993          6095  2     ! cancelled.  Then resignal the error.
;  5994          6096  2     !
;  5995          6097  3     IF (.SIGARG[1] EQL DBG$_NOFREE) OR (.SIGARG[1] EQL DBG$_ABORTED)
;  5996          6098  2     THEN
;  5997          6099  3         BEGIN
;  5998          6100  3         RSTPTR = .MOD_BEING_BUILT;
;  5999          6101  3         RSTPTR[RST$V_MODSET] = FALSE;
;  6000          6102  3         DBG$RST_REMOVE(.MOD_BEING_BUILT);
;  6001          6103  2         END;
;  6002          6104  2
;  6003          6105  2     RETURN SS$_RESIGNAL;
;  6004          6106  2
;  6005          6107  1     END;


                                0004 00000 RST_BUILD_ERROR_HANDLER:
                                           .WORD   Save R2                                 ; 6061
            52 00000000'   EF   9E 00002           MOVAB   MOD_BEING_BUILT, R2             ;
            50       04    AC   D0 00009           MOVL    SIGARG, R0                      ; 6097
00028332    8F       04    A0   D1 0000D           CMPL    4(R0), #164658                  ;
```

```
                                 0A  13 00015           BEQL    1$                                      ;
                 000280E8    8F  04  A0  D1 00017           CMPL    4(R0), #164072                          ;
                                 0E  12 0001F           BNEQ    2$                                      ;
                             50      62  D0 00021 1$:    MOVL    MOD_BEING_BUILT, RSTPTR                 ; 6100
                          28 A0      01  8A 00024           BICB2   #1, 40(RSTPTR)                  ; 6101
                                     62  DD 00028           PUSHL   MOD_BEING_BUILT                 ; 6102
                        F79E CF      01  FB 0002A           CALLS   #1, DBG$RST_REMOVE              ;
                             50  0918 8F  3C 0002F 2$:    MOVZWL  #2328, R0                       ; 6105
                                     04  00034           RET                                     ; 6107

; Routine Size:  53 bytes,    Routine Base:  DBG$CODE + 2C9B
```

```
;  6007          6108  1 ROUTINE SHARED_IMAGE_NAMES(IMGFILCHAN): NOVALUE =
;  6008          6109  1 !
;  6009          6110  1 ! FUNCTION
;  6010          6111  1 !
;  6011          6112  1 !       ***** This routine is used for VMS 4.0 (3B) shareable image format *****
;  6012          6113  1 !
;  6013          6114  1 !       This routine goes through the list of image names in the current
;  6014          6115  1 !       execution and creates a Global Symbol for each such image name.
;  6015          6116  1 !       The images in the current execution include the user program, all
;  6016          6117  1 !       sharable images used by the user program, and even DEBUG itself.
;  6017          6118  1 !       For each image name xxx, this routine creates an RST entry for
;  6018          6119  1 !       the symbol SHARE$xxx, marks that RST entry as being a global symbol,
;  6019          6120  1 !       and attaches it to the global symbol chain off the anonymous module.
;  6020          6121  1 !       This is intended to be an aid to programmers debugging shared im-
;  6021          6122  1 !       ages since it provides a way of telling where a shared image starts.
;  6022          6123  1 !
;  6023          6124  1 !       The sharable image names and their addresses are gotten from the
;  6024          6125  1 !       Sharable Image Descriptor List Elements in the Image Descriptor Blocks
;  6025          6126  1 !       in P1 space.  The external symbol IAC$GL_IMAGE_LIST is used to access
;  6026          6127  1 !       the doubly linked list of Image Descriptor Blocks; it points to a
;  6027          6128  1 !       2 longword vector in which the 1st longword points to the first
;  6028          6129  1 !       Descriptor Block on the list. Note - for FT2, a symbol called
;  6029          6130  1 !       SYS$GL_IMGLSTPTR was put in STARLET which we use instead of
;  6030          6131  1 !       IAC$GL_IMAGE_LIST.
;  6031          6132  1 !
;  6032          6133  1 !       The Image Descriptor Blocks do not contain duplicate entries for
;  6033          6134  1 !       the same sharable image.  This code assumes this is true, and no check
;  6034          6135  1 !       is done to validate it.
;  6035          6136  1 !
;  6036          6137  1 !       This routine also creates Module RST entry for each shareable
;  6037          6138  1 !       image.
;  6038          6139  1 !
;  6039          6140  1 ! INPUTS
;  6040          6141  1 !       IMGFILCHAN - The channel that the image file is open on.
;  6041          6142  1 !
;  6042          6143  1 ! OUTPUTS
;  6043          6144  1 !       NONE
;  6044          6145  1 !
;  6045          6146  1
;  6046          6147  2     BEGIN
;  6047          6148  2
;  6048          6149  2     LOCAL
;  6049          6150  2         DSTPTR: REF DST$RECORD,          ! Pointer to Module DST pointer
;  6050          6151  2         GBLNAME: VECTOR[                 ! Buffer in which we build the global
;  6051          6152  2                 IMDB$C_MAXNAMLNG+7,BYTE],!       symbol name "SHARE$xxx"
;  6052          6153  2         HEADER,
;  6053          6154  2         IMDBPTR: REF BLOCK[,BYTE],       ! Pointer to Image Descriptor Block
;  6054          6155  2         LENGTH,                          ! Length of the Module DST
;  6055          6156  2         NAMEPTR: REF VECTOR[,BYTE],      ! Pointer to shared image name "xxx"
;  6056          6157  2         RSTPTR: REF RST$ENTRY;           ! Pointer to duplicate RST entry
;  6057          6158  2
;  6058          6159  2
;  6059          6160  2     ! Enable a handler that will catch any errors that might occur
;  6060          6161  2     ! during this routine. This is mainly done as a defensive move
;  6061          6162  2     ! in case the symbols IAC$GL_IMAGE_LIST or SYS$GL_IMGLSTPTR
;  6062          6163  2     ! are incorrect or the data structures they point to are incorrect
;  6063          6164  2     ! (e.g., as a result of image activator changes). In that case,
```

```
; 6064          6165  2     ! we signal an informational to the effect that the SHARE$xxx names
; 6065          6166  2     ! were not available, but we do not want to abort the debugging session
; 6066          6167  2     ! (which is what would happen without a handler).
; 6067          6168  2     !
; 6068          6169  2     ENABLE
; 6069          6170  2         SHARED_IMAGE_NAMES_HANDLER;
; 6070          6171  2
; 6071          6172  2
; 6072          6173  2     ! Get the start address of the Image Descriptor Block list.  Then loop
; 6073          6174  2     ! through this doubly linked list of Descriptor Blocks until we get to the
; 6074          6175  2     ! head of the list, and for each Descriptor Block, make global symbols for
; 6075          6176  2     ! any image entries in the block.
; 6076          6177  2     !
; 6077          6178  2     IF SYS$GL_IMGLSTPTR NEQ 0
; 6078          6179  2     THEN
; 6079          6180  2         IMDBPTR = .SYS$GL_IMGLSTPTR
; 6080          6181  2     ELSE IF IAC$GL_IMAGE_LIST NEQ 0
; 6081          6182  2     THEN
; 6082          6183  2         IMDBPTR = IAC$GL_IMAGE_LIST
; 6083          6184  2     ELSE
; 6084          6185  2         SIGNAL(0);      ! The handler will catch it.
; 6085          6186  2     HEADER = .IMDBPTR;
; 6086          6187  2     IMDBPTR = .IMDBPTR[IMDB$L_FLINK];
; 6087          6188  2     WHILE .IMDBPTR NEQ .HEADER DO
; 6088          6189  3         BEGIN
; 6089          6190  3
; 6090          6191  3
; 6091          6192  3         ! For each sharable image, create the corresponding global sym-
; 6092          6193  3         ! bol by prefixing the image name with "SHARE$".
; 6093          6194  3         !
; 6094          6195  3         NAMEPTR = IMDBPTR[IMDB$T_IMAGE_NAME];
; 6095          6196  3         CH$MOVE(6, UPLIT BYTE(%ASCII 'SHARE$'), GBLNAME[1]);
; 6096          6197  3         CH$MOVE(.NAMEPTR[0], NAMEPTR[1], GBLNAME[7]);
; 6097          6198  3         GBLNAME[0] = .NAMEPTR[0] + 6;
; 6098          6199  3
; 6099          6200  3
; 6100          6201  3         ! Create a Global symbol RST entry for the new symbol.
; 6101          6202  3         ! Ignore the main user exe.
; 6102          6203  3         !
; 6103          6204  3         IF .IMDBPTR[IMDB$W_CHAN] NEQ .IMGFILCHAN
; 6104          6205  3         THEN
; 6105          6206  4             BEGIN
; 6106          6207  4
; 6107          6208  4
; 6108          6209  4             ! Enter this symbol into Global symbol chain.
; 6109          6210  4             !
; 6110          6211  4             DBG$GST_TO_RST(.RST$START_ADDR, GBLNAME[0],
; 6111          6212  4                 .IMDBPTR[IMDB$L_STARTING_ADDRESS], FALSE);
; 6112          6213  4
; 6113          6214  4
; 6114          6215  4             ! 1 byte of DST length, 1 byte of DST kind, 1 byte of
; 6115          6216  4             ! MODBEG_UNUSED, 4 bytes of MODBEG_LANGUAGE, and a
; 6116          6217  4             ! counted ASCIC string.
; 6117          6218  4             !
; 6118          6219  4             LENGTH = 1 + 1 + 1 + 4 + 1 + .GBLNAME[0];
; 6119          6220  4
; 6120          6221  4
```

```
; 6121          6222  4          ! Create Module RST Entry in the module RST chain for this shareable
; 6122          6223  4          ! image.
; 6123          6224  4          !
; 6124          6225  4          RSTPTR = GET_MEM(RST$K_SHARED_MODENTSIZ + (.LENGTH + 3) / %UPVAL);
; 6125          6226  4          RSTPTR[RST$L_BASEVA] = .IMDBPTR[IMDB$L_STARTING_ADDRESS];
; 6126          6227  4          RSTPTR[RST$B_IMGFILCHAN] = .IMDBPTR[IMDB$W_CHAN];
; 6127          6228  4          RSTPTR[RST$V_SHARE_IMAGE] = 1;
; 6128          6229  4
; 6129          6230  4
; 6130          6231  4          ! Create a dummy Module Begin DST record for the shareable image
; 6131          6232  4          ! "module".
; 6132          6233  4          !
; 6133          6234  4          DSTPTR = .RSTPTR + %UPVAL * RST$K_SHARED_MODENTSIZ;
; 6134          6235  4          DSTPTR[DST$B_LENGTH] = .LENGTH - 1;
; 6135          6236  4          DSTPTR[DST$B_TYPE] = DST$K_MODBEG;
; 6136          6237  4          DSTPTR[DST$L_MODBEG_LANGUAGE] = DBG$K_UNKNOWN;
; 6137          6238  4          NAMEPTR = DSTPTR[DST$B_MODBEG_NAME];
; 6138          6239  4          NAMEPTR[0] = .GBLNAME[0];
; 6139          6240  4          CH$MOVE(.NAMEPTR[0], GBLNAME[1], NAMEPTR[1]);
; 6140          6241  4          RSTPTR[RST$L_DSTPTR] = .DSTPTR;
; 6141          6242  4          RSTPTR[RST$B_KIND] = RST$K_MODULE;
; 6142          6243  4          RSTPTR[RST$B_LANGUAGE] = DBG$K_UNKNOWN;
; 6143          6244  4          DBG$HASH_INSERT(.RSTPTR);
; 6144          6245  4
; 6145          6246  4
; 6146          6247  4          ! Build the Program Static Address entry for this module.
; 6147          6248  4          !
; 6148          6249  4          LENGTH =
; 6149          6250  4              .IMDBPTR[IMDB$L_END_ADDRESS] - .IMDBPTR[IMDB$L_STARTING_ADDRESS] + 1;
; 6150          6251  4          STATIC_ADDRESS(.RSTPTR[RST$L_BASEVA], .LENGTH, .RSTPTR,
; 6151          6252  4                         SAT$START_ADDR, FALSE);
; 6152          6253  4
; 6153          6254  4
; 6154          6255  4          ! Update the next module chain pointer.
; 6155          6256  4          !
; 6156          6257  4          PREV_RST_PTR[RST$L_NXTMODPTR] = .RSTPTR;
; 6157          6258  4          PREV_RST_PTR = .RSTPTR;
; 6158          6259  3          END;
; 6159          6260  3
; 6160          6261  3
; 6161          6262  3      ! Link to the next Image Descriptor Block and loop.
; 6162          6263  3      !
; 6163          6264  3      IMDBPTR = .IMDBPTR[IMDB$L_FLINK];
; 6164          6265  2      END;
; 6165          6266  2
; 6166          6267  2
; 6167          6268  2  ! We have now gone through all Descriptor Blocks.  Return to the caller.
; 6168          6269  2  !
; 6169          6270  2  RETURN;
; 6170          6271  2
; 6171          6272  1  END;


                                                    .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                    24  45  52  41  48  53  0017D P.AAW:  .ASCII  \SHARE$\                          ;
```

```
                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                OFFC 00000 SHARED_IMAGE_NAMES:
                                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 6108
                                     5E     FF78    CE   9E 00002          MOVAB   -136(SP), SP
                                     6D     00FE    CF   DE 00007          MOVAL   6$, (FP)                    ; 6147
                                     50 00000000G   00   9E 0000C          MOVAB   SYS$GL_IMGLSTPTR, R0        ; 6178
                                                    09   13 00013          BEQL    1$
                                     57 00000000G   00   D0 00015          MOVL    SYS$GL_IMGLSTPTR, IMDBPTR   ; 6180
                                                    1B   11 0001C          BRB     3$
                                     50 00000000G   00   9E 0001E 1$:       MOVAB   IAC$GL_IMAGE_LIST, R0       ; 6181
                                                    09   13 00025          BEQL    2$
                                     57 00000000G   00   9E 00027          MOVAB   IAC$GL_IMAGE_LIST, IMDBPTR  ; 6183
                                                    09   11 0002E          BRB     3$
                                                    7E   D4 00030 2$:       CLRL    -(SP)                       ; 6185
                                          00000000G 00                      01   FB 00032          CALLS   #1, LIB$SIGNAL
                                                    5B                      57   D0 00039 3$:       MOVL    IMDBPTR, HEADER             ; 6186
                                                    57                      67   D0 0003C 4$:       MOVL    (IMDBPTR), IMDBPTR          ; 6187
                                                    5B                      57   D1 0003F           CMPL    IMDBPTR, HEADER             ; 6188
                                                                            01   12 00042          BNEQ    5$
                                                                                 04 00044          RET
                                     58         14      A7   9E 00045 5$:       MOVAB   20(R7), NAMEPTR             ; 6195
                                01   AE 00000000'       EF                      06   28 00049          MOVC3   #6, P.AAW, GBLNAME+1        ; 6196
                                     50                                         68   9A 00052          MOVZBL  (NAMEPTR), R0               ; 6197
                                07   AE         01      A8                      50   28 00055          MOVC3   R0, 1(NAMEPTR), GBLNAME+7
                                     6E                                         68   06 81 0005B       ADDB3   #6, (NAMEPTR), GBLNAME      ; 6198
                    04   AC     0E   A7                 10                      00   ED 0005F          CMPZV   #0, #16, 14(IMDBPTR), IMGFILCHAN    ; 6204
                                                                                D4   13 00066          BEQL    4$
                                                                                7E   D4 00068          CLRL    -(SP)                       ; 6211
                                                                                          48   A7   DD 0006A  PUSHL   72(IMDBPTR)                  ; 6212
                                                                                          08   AE   9F 0006D  PUSHAB  GBLNAME                     ; 6211
                                                                                00000000'      EF   DD 00070  PUSHL   RST$START_ADDR
                                                                                D818   CF             04   FB 00076  CALLS   #4, DBG$GST_TO_RST
                                                                                       5A             6E   9A 0007B  MOVZBL  GBLNAME, LENGTH         ; 6219
                                                                                       5A             08   C0 0007E  ADDL2   #8, LENGTH
                                                                                       50         03  AA   9E 00081  MOVAB   3(R10), R0            ; 6225
                                                                                       50             04   C6 00085  DIVL2   #4, R0
                                                                                                  0E   A0   9F 00088  PUSHAB  14(R0)
                                                                                FB43   CF             01   FB 0008B  CALLS   #1, GET_MEM
                                                                                       56             50   D0 00090  MOVL    R0, RSTPTR
                                                                                   30  A6         48  A7   D0 00093  MOVL    72(IMDBPTR), 48(RSTPTR)     ; 6226
                                                                                   34  A6         0E  A7   90 00098  MOVB    14(IMDBPTR), 52(RSTPTR)     ; 6227
                                                                                   28  A6             10   88 0009D  BISB2   #16, 40(RSTPTR)          ; 6228
                                                                                       59         38  A6   9E 000A1  MOVAB   56(R6), DSTPTR          ; 6234
                                                                              69                   5A             01   83 000A5  SUBB3   #1, LENGTH, (DSTPTR)    ; 6235
                                                                                   01  A9         BC  8F   90 000A9  MOVB    #-68, 1(DSTPTR)             ; 6236
                                                                                   03  A9             0A   D0 000AE  MOVL    #10, 3(DSTPTR)              ; 6237
                                                                                       58         07  A9   9E 000B2  MOVAB   7(R9), NAMEPTR          ; 6238
                                                                                       68             6E   90 000B6  MOVB    GBLNAME, (NAMEPTR)          ; 6239
                                                                                       50             68   9A 000B9  MOVZBL  (NAMEPTR), R0               ; 6240
                                                                              01   A8   01  AE             50   28 000BC  MOVC3   R0, GBLNAME+1, 1(NAMEPTR)
                                                                                   0C  A6             59   D0 000C2  MOVL    DSTPTR, 12(RSTPTR)          ; 6241
                                                                                   14  A6             01   90 000C6  MOVB    #1, 20(RSTPTR)              ; 6242
                                                                                   29  A6             0A   90 000CA  MOVB    #10, 41(RSTPTR)             ; 6243
```

```
                                    56  DD 000CE          PUSHL   RSTPTR                                          ; 6244
                          D997      CF     01  FB 000D0  CALLS   #1, DBG$HASH_INSERT                            ;
                    50      4C      A7     48  A7 C3 000D5  SUBL3   72(IMDBPTR), -76(IMDBPTR), R0               ; 6250
                                    5A     01  A0 9E 000DB  MOVAB   1(R0), LENGTH                               ;
                                           7E  D4 000DF      CLRL    -(SP)                                       ; 6251
                                00000000'  EF  9F 000E1      PUSHAB  SAT$START_ADDR                              ;
                                           56  DD 000E7      PUSHL   RSTPTR                                      ;
                                           5A  DD 000E9      PUSHL   LENGTH                                      ;
                                       30  A6  DD 000EB      PUSHL   48(RSTPTR)                                  ;
                                0000V  CF     05  FB 000EE  CALLS   #5, STATIC_ADDRESS                          ;
                                    50 00000000'  EF  D0 000F5  MOVL    PREV_RST_PTR, R0                        ; 6257
                                   10   A0          56  D0 000FA  MOVL    RSTPTR, 16(R0)                      ;
                           00000000'   EF          56  D0 000FE  MOVL    RSTPTR, PREV_RST_PTR                ; 6258
                                    FF34   31 00105  BRW     4$                                                 ; 6264
                                           04 00108  RET                                                       ; 6272
                                      0000 00109 6$:  .WORD   Save nothing                                      ; 6147
                                           7E  D4 0010B      CLRL    -(SP)                                       ;
                                           5E  DD 0010D      PUSHL   SP                                          ;
                                    7E      04  AC 7D 0010F  MOVQ    4(AP), -(SP)                                ;
                                0000V  CF     03  FB 00113  CALLS   #3, SHARED_IMAGE_NAMES_HANDLER              ;
                                           04 00118  RET                                                       ;

; Routine Size:  281 bytes,    Routine Base:  DBG$CODE + 2CD0
```

```
;  6173          6273  1 ROUTINE SHARED_IMAGE_NAMES_HANDLER(SIGNAL_ARG_PTR: REF BLOCK[,BYTE]) =
;  6174          6274  1 !
;  6175          6275  1 ! FUNCTION
;  6176          6276  1 !
;  6177          6277  1 !       This routine is the error handler for the SHARED_IMAGE_NAMES
;  6178          6278  1 !       routine.
;  6179          6279  1 !
;  6180          6280  1 !       An error handler is used as a defensive move
;  6181          6281  1 !       in case the symbol IAC$GL_IMAGE_LIST is incorrect or the data
;  6182          6282  1 !       structures it points to are incorrect (e.g., as a result of
;  6183          6283  1 !       image activator changes). In that case, we want to signal an
;  6184          6284  1 !       informational to the effect that the SHARE$xxx names were not
;  6185          6285  1 !       available, but we do not want to abort the debugging session
;  6186          6286  1 !       (which is what would happen without a handler).
;  6187          6287  1 !
;  6188          6288  1 ! INPUTS
;  6189          6289  1 !       SIGNAL_ARG_PTR points to the signal arguments for the exception.
;  6190          6290  1 !
;  6191          6291  1 ! OUTPUTS
;  6192          6292  1 !       This routine unwinds the stack to the caller of SHARED_IMAGE_NAMES
;  6193          6293  1 !       and then returns SS$_CONTINUE.
;  6194          6294  1 !
;  6195          6295  2     BEGIN
;  6196          6296  2
;  6197          6297  2
;  6198          6298  2     ! If we get into this handler for the second time (because of
;  6199          6299  2     ! the unwind), then resignal.
;  6200          6300  2     !
;  6201          6301  2     IF .SIGNAL_ARG_PTR[CHF$L_SIG_NAME] EQL SS$_UNWIND
;  6202          6302  2     THEN
;  6203          6303  2         RETURN SS$_RESIGNAL;
;  6204          6304  2
;  6205          6305  2
;  6206          6306  2     ! Put out an informational saying that the shared image names
;  6207          6307  2     ! are not available. We fake the message since the final
;  6208          6308  2     ! handler is not set up properly to accept a signal here.
;  6209          6309  2     ! Only do this for V4.
;  6210          6310  2     !
;  6211          6311  2     IF .DBG$GV_CONTROL[DBG$V_CONTROL_VERSION_4]
;  6212          6312  2     THEN
;  6213          6313  3         BEGIN
;  6214          6314  3         DBG$PRINT(UPLIT BYTE(%ASCIC
;  6215          6315  3             '%DEBUG-I-ERRSHRNAM, internal error during initialization, SHARE$ symbols not available'));
;  6216          6316  3         DBG$NEWLINE();
;  6217          6317  2         END;
;  6218          6318  2
;  6219          6319  2
;  6220          6320  2     ! Unwind the stack to the caller of SHARED_IMAGE_NAMES and then
;  6221          6321  2     ! return SS$_CONTINUE. What this does is abort processing of
;  6222          6322  2     ! SHARED_IMAGE_NAMES and allow the rest of DEBUG initialization
;  6223          6323  2     ! to proceed.
;  6224          6324  2     !
;  6225          6325  2     $SETUNWIND();
;  6226          6326  2     RETURN SS$_CONTINUE;
;  6227          6327  1     END;
```

```
                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

48 53 52 52 45 2D 49 2D 47 55 42 45 44 25 56  00183 P.AAX:   .ASCII  \V%DEBUG-I-ERRSHRNAM, internal error duri\  ;
20 6C 61 6E 72 65 74 6E 69 20 2C 4D 41 4E 52  00192                                                           ;
                69 72 75 64 20 72 6F 72 72 65  001A1                                                           ;
69 74 61 7A 69 6C 61 69 74 69 6E 69 20 67 6E  001AB           .ASCII  \ng initialization, SHARE$ symbols not av\  ;
62 6D 79 73 20 24 45 52 41 48 53 20 2C 6E 6F  001BA                                                           ;
                76 61 20 74 6F 6E 20 73 6C 6F  001C9                                                           ;
                      65 6C 62 61 6C 69 61  001D3           .ASCII  \ailable\                               ;


                                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                0000 00000 SHARED_IMAGE_NAMES_HANDLER:
                                                           .WORD   Save nothing                                 ; 6273
                          50        04     AC  D0 00002           MOVL    SIGNAL_ARG_PTR, R0                   ; 6301
                00000920  8F        04     A0  D1 00006           CMPL    4(R0), #2356                         ;
                                           06  12 0000E           BNEQ    1$                                   ;
                          50      0918     8F  3C 00010           MOVZWL  #2328, R0                            ; 6303
                                           04  00015           1$: RET                                          ;
                14 00000000G      00       04  E1 00016           BBC     #4, DBG$GV_CONTROL+1, 2$             ; 6311
                                  00000000' EF  9F 0001E           PUSHAB  P.AAX                                ; 6314
                00000000G         00        01  FB 00024           CALLS   #1, DBG$PRINT                        ;
                00000000G         00        00  FB 0002B           CALLS   #0, DBG$NEWLINE                      ; 6316
                                            7E  7C 00032 2$:        CLRQ    -(SP)                                ; 6325
                00000000G         00        02  FB 00034           CALLS   #2, SYS$UNWIND                       ;
                          50                01  D0 0003B           MOVL    #1, R0                               ; 6326
                                            04  0003E           1$: RET                                          ; 6327

; Routine Size:  63 bytes,    Routine Base:  DBG$CODE + 2DE9
```

```
;  6229          6328  1 ROUTINE SHARED_IMAGE_NAMES_V3(IMGFILCHAN): NOVALUE =
;  6230          6329  1 !
;  6231          6330  1 !  FUNCTION
;  6232          6331  1 !
;  6233          6332  1 !       ***** This routine is used for VMS 3.0 shareable image format *****
;  6234          6333  1 !
;  6235          6334  1 !       This routine goes through the list of image names in the current
;  6236          6335  1 !       execution and creates a Global Symbol for each such image name.
;  6237          6336  1 !       The images in the current execution include the user program, all
;  6238          6337  1 !       sharable images used by the user program, and even DEBUG itself.
;  6239          6338  1 !       For each image name xxx, this routine creates an RST entry for
;  6240          6339  1 !       the symbol SHARE$xxx, marks that RST entry as being a global symbol,
;  6241          6340  1 !       and attaches it to the global symbol chain off the anonymous module.
;  6242          6341  1 !       This is intended to be an aid to programmers debugging shared im-
;  6243          6342  1 !       ages since it provides a way of telling where a shared image starts.
;  6244          6343  1 !
;  6245          6344  1 !       The sharable image names and their addresses are gotten from the
;  6246          6345  1 !       Sharable Image List Elements in the Image Activator Fixup Blocks
;  6247          6346  1 !       in P1 space.  The external symbol SYS$GL_IAFLINK is used to access
;  6248          6347  1 !       the linked list of Image Activator Fixup Blocks; it points to a
;  6249          6348  1 !       pointer to the first Fixup Block on the list.
;  6250          6349  1 !
;  6251          6350  1 !       The Image Activator Fixup Blocks may contain duplicate entries for
;  6252          6351  1 !       the same sharable image.  This happens when two sharable images in
;  6253          6352  1 !       the execution reference a third image, for example--in this case
;  6254          6353  1 !       there are two entries for that third image.  This code assumes
;  6255          6354  1 !       that only the first such entry is valid and ignores all succeeding
;  6256          6355  1 !       duplicates.  If the image activator is ever changed to invalidate
;  6257          6356  1 !       this assumption, this routine could break.
;  6258          6357  1 !
;  6259          6358  1 !       This routine also creates Module RST entry for each shareable
;  6260          6359  1 !       image.
;  6261          6360  1 !
;  6262          6361  1 !  INPUTS
;  6263          6362  1 !       IMGFILCHAN - The channel that the image file is open on.
;  6264          6363  1 !
;  6265          6364  1 !  OUTPUTS
;  6266          6365  1 !       NONE
;  6267          6366  1 !
;  6268          6367  1
;  6269          6368  2     BEGIN
;  6270          6369  2
;  6271          6370  2     LOCAL
;  6272          6371  2         CHANNEL_NUMBER,                 ! The channel number that the image
;  6273          6372  2                                         !      file is open on
;  6274          6373  2         DSTPTR: REF DST$RECORD,         ! Pointer to Module dst pointer
;  6275          6374  2         GBLNAME: VECTOR[                ! Buffer in which we build the global
;  6276          6375  2                 SHL$C_MAXNAMLNG+7,BYTE],!      symbol name "SHARE$xxx"
;  6277          6376  2         IAFPTR: REF BLOCK[,BYTE],       ! Pointer to Image Activator Fixup Block
;  6278          6377  2         LENGTH,                         ! Length of the Module DST
;  6279          6378  2         NAMEPTR: REF VECTOR[,BYTE],     ! Pointer to shared image name "xxx"
;  6280          6379  2         RSTPTR: REF RST$ENTRY,          ! Pointer to duplicate RST entry
;  6281          6380  2         SHLPTR: REF BLOCKVECTOR[        ! Pointer to Sharable Image List vector
;  6282          6381  2                 ,SHL$K_LENGTH,BYTE];    !
;  6283          6382  2
;  6284          6383  2
;  6285          6384  2
```

```
; 6286          6385  2     ! Enable a handler that will catch any errors that might occur
; 6287          6386  2     ! during this routine. This is mainly done as a defensive move
; 6288          6387  2     ! in case the symbol IAC$GL_IMAGE_LIST is incorrect or the data
; 6289          6388  2     ! structures it points to are incorrect (e.g., as a result of
; 6290          6389  2     ! image activator changes). In that case, we want to signal an
; 6291          6390  2     ! informational to the effect that the SHARE$xxx names were not
; 6292          6391  2     ! available, but we do not want to abort the debugging session
; 6293          6392  2     ! (which is what would happen without a handler).
; 6294          6393  2     !
; 6295          6394  2     ENABLE
; 6296          6395  2         SHARED_IMAGE_NAMES_HANDLER;
; 6297          6396  2
; 6298          6397  2
; 6299          6398  2     ! Get the start address of the Image Activator Fixup Block list.  Then loop
; 6300          6399  2     ! through this linked list of Fixup Blocks until we get to the end of the
; 6301          6400  2     ! list, and for each Fixup Block, make global symbols for any image entries
; 6302          6401  2     ! in the block.
; 6303          6402  2     !
; 6304          6403  2     IAFPTR = .SYS$GL_IAFLINK[0];
; 6305          6404  2     WHILE .IAFPTR NEQ 0 DO
; 6306          6405  3         BEGIN
; 6307          6406  3         SHLPTR = .IAFPTR + .IAFPTR[IAF$L_SHLSTOFF];
; 6308          6407  3
; 6309          6408  3
; 6310          6409  3         ! Loop through the vector of Sharable Image List Elements in this Fixup
; 6311          6410  3         ! Block.  For each sharable image, create the corresponding global sym-
; 6312          6411  3         ! bol by prefixing the image name with "SHARE$".
; 6313          6412  3         !
; 6314          6413  3         INCR I FROM 0 TO .IAFPTR[IAF$L_SHRIMGCNT] - 1 DO
; 6315          6414  4             BEGIN
; 6316          6415  4
; 6317          6416  4
; 6318          6417  4             ! Create the new symbol name by prefixing SHARE$ to the image name.
; 6319          6418  4             !
; 6320          6419  4             NAMEPTR = SHLPTR[.I, SHL$T_IMGNAM];
; 6321          6420  4             CH$MOVE(6, UPLIT BYTE(%ASCII 'SHARE$'), GBLNAME[1]);
; 6322          6421  4             CH$MOVE(.NAMEPTR[0], NAMEPTR[1], GBLNAME[7]);
; 6323          6422  4             GBLNAME[0] = .NAMEPTR[0] + 6;
; 6324          6423  4
; 6325          6424  4
; 6326          6425  4             ! Get the channel number, now the channel number is carried in
; 6327          6426  4             ! the last byte of the shared image NAME.  This may change
; 6328          6427  4             ! in the future.
; 6329          6428  4             !
; 6330          6429  4             CHANNEL_NUMBER = .NAMEPTR[30];
; 6331          6430  4
; 6332          6431  4
; 6333          6432  4             ! Determine whether we already have a Global symbol RST entry for
; 6334          6433  4             ! the SHARE$xxx symbol.  If so, this is a duplicate Fixup Block
; 6335          6434  4             ! entry and we ignore it.  Otherwise, create a Global symbol RST
; 6336          6435  4             ! entry for the new symbol.
; 6337          6436  4             !
; 6338          6437  4             DBG$HASH_FIND_SETUP(GBLNAME);
; 6339          6438  4             WHILE TRUE DO
; 6340          6439  5                 BEGIN
; 6341          6440  5                 RSTPTR = DBG$HASH_FIND(GBLNAME);
; 6342          6441  5                 IF .RSTPTR EQL 0
```

```
;  6343          6442  5                THEN
;  6344          6443  6                    BEGIN
;  6345          6444  6                    IF .CHANNEL_NUMBER NEQ .IMGFILCHAN
;  6346          6445  6                    THEN
;  6347          6446  7                        BEGIN
;  6348          6447  7
;  6349          6448  7
;  6350          6449  7                        ! Enter this symbol into Global symbol chain.
;  6351          6450  7                        !
;  6352          6451  7                        DBG$GST_TO_RST(.RST$START_ADDR, GBLNAME[0], .SHLPTR[.I,SHL$L_BASEVA], FALSE);
;  6353          6452  7
;  6354          6453  7
;  6355          6454  7                        ! Create module rst entry in the module rst chain.
;  6356          6455  7                        !
;  6357          6456  7                        RSTPTR = GET_MEM(RST$K_SHARED_MODENTSIZ);
;  6358          6457  7                        RSTPTR[RST$L_BASEVA] = .SHLPTR[.I, SHL$L_BASEVA];
;  6359          6458  7                        RSTPTR[RST$B_IMGFILCHAN] = .CHANNEL_NUMBER;
;  6360          6459  7                        RSTPTR[RST$V_SHARE_IMAGE] = 1;
;  6361          6460  7
;  6362          6461  7
;  6363          6462  7                        ! 1 byte of DST length, 1 byte of DST kind, 1 byte of
;  6364          6463  7                        ! MODBEG_UNUSED, 4 bytes of MODBEG_LANGUAGE, and
;  6365          6464  7                        ! counted ASCIC string.
;  6366          6465  7                        !
;  6367          6466  7                        LENGTH = 1 + 1 + 1 + 4 + 1 + .GBLNAME[0];
;  6368          6467  7                        DSTPTR = DBG$GET_MEMORY((.LENGTH + 3) / 4);
;  6369          6468  7                        DSTPTR[DST$B_LENGTH] = .LENGTH - 1;
;  6370          6469  7                        DSTPTR[DST$B_TYPE] = DST$K_MODBEG;
;  6371          6470  7                        DSTPTR[DST$L_MODBEG_LANGUAGE] = DBG$K_UNKNOWN;
;  6372          6471  7                        NAMEPTR = DSTPTR[DST$B_MODBEG_NAME];
;  6373          6472  7                        NAMEPTR[0] = .GBLNAME[0];
;  6374          6473  7                        CH$MOVE(.NAMEPTR[0], GBLNAME[1], NAMEPTR[1]);
;  6375          6474  7                        RSTPTR[RST$L_DSTPTR] = .DSTPTR;
;  6376          6475  7                        RSTPTR[RST$B_KIND] = RST$K_MODULE;
;  6377          6476  7                        RSTPTR[RST$B_LANGUAGE] = DBG$K_UNKNOWN;
;  6378          6477  7                        DBG$HASH_INSERT(.RSTPTR);
;  6379          6478  7
;  6380          6479  7
;  6381          6480  7                        ! Build the Program Static Address entry for this
;  6382          6481  7                        ! module.
;  6383          6482  7                        !
;  6384          6483  7                        STATIC_ADDRESS(.RSTPTR[RST$L_BASEVA], 1, .RSTPTR,
;  6385          6484  7                                SAT$START_ADDR, FALSE);
;  6386          6485  7
;  6387          6486  7
;  6388          6487  7                        ! Update the next module chain pointer.
;  6389          6488  7                        !
;  6390          6489  7                        PREV_RST_PTR[RST$L_NXTMODPTR] = .RSTPTR;
;  6391          6490  7                        PREV_RST_PTR = .RSTPTR;
;  6392          6491  6                        END;
;  6393          6492  6
;  6394          6493  6                    EXITLOOP;
;  6395          6494  5                    END;
;  6396          6495  5
;  6397          6496  5                IF .RSTPTR[RST$V_GLOBAL] THEN EXITLOOP;
;  6398          6497  4                END;
;  6399          6498  4
```

```
;  6400          6499  3          END;
;  6401          6500  3
;  6402          6501  3
;  6403          6502  3          ! Link to the next Image Activator Fixup Block and loop.
;  6404          6503  3          !
;  6405          6504  3          IAFPTR = .IAFPTR[IAF$L_IAFLINK];
;  6406          6505  2          END;
;  6407          6506  2
;  6408          6507  2
;  6409          6508  2      ! We have now gone through all Fixup Blocks.  Return to the caller.
;  6410          6509  2      !
;  6411          6510  2      RETURN;
;  6412          6511  2
;  6413          6512  1      END;


                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                        24  45  52  41  48  53  001DA P.AAY:  .ASCII  \SHARE$\                                  ;


                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                          0FFC 00000 SHARED_IMAGE_NAMES_V3:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 6328
                            5E                3C  C2 00002        SUBL2   #60, SP                             ;
                            6D      0128      CF  DE 00005        MOVAL   11$, (FP)                           ; 6368
                            50 00000000G      00  D0 0000A        MOVL    SYS$GL_IAFLINK, R0                  ; 6403
                        08  AE                60  D0 00011        MOVL    (R0), IAFPTR                        ;
                                              01  12 00015 1$:    BNEQ    2$                                  ; 6404
                                                  04 00017        RET                                         ;
                    50      08  AE                18  C1 00018 2$:    ADDL3   #24, IAFPTR, R0                 ; 6406
                    59      08  AE                60  C1 0001D        ADDL3   (R0), IAFPTR, SHLPTR            ;
                                                  56  01  CE 00022    MNEGL   #1, I                           ; 6419
                                              00F5  31 00025 3$:    BRW     8$                                  ;
                    5B                        56      06  78 00028 4$:    ASHL    #6, I, R11                  ;
                                              5A  18 AB49  9E 0002C        MOVAB   24(R11)[SHLPTR], NAMEPTR    ;
                0D  AE  00000000'             EF      06  28 00031        MOVC3   #6, P.AAY, GBLNAME+1        ; 6420
                                              50      6A  9A 0003A        MOVZBL  (NAMEPTR), R0               ; 6421
                13  AE          01            AA      50  28 0003D        MOVC3   R0, 1(NAMEPTR), GBLNAME+7   ;
                0C  AE                        6A      06  81 00043        ADDB3   #6, (NAMEPTR), GBLNAME      ; 6422
                                              6E  1E  AA  9A 00048        MOVZBL  30(NAMEPTR), CHANNEL_NUMBER ; 6429
                                            0C  AE  9F  0004C        PUSHAB  GBLNAME                         ; 6437
                                  D862      CF      01  FB 0004F        CALLS   #1, DBG$HASH_FIND_SETUP     ;
                                            0C  AE  9F  00054 5$:    PUSHAB  GBLNAME                         ; 6440
                                  D7DD      CF      01  FB 00057        CALLS   #1, DBG$HASH_FIND           ;
                                              58      50  D0 0005C        MOVL    R0, RSTPTR                  ;
                                                      03  13 0005F        BEQL    6$                          ; 6441
                                                  00B2  31 00061        BRW     7$                          ;
                            04  AC                6E  D1 00064 6$:    CMPL    CHANNEL_NUMBER, IMGFILCHAN  ; 6444
                                                  BB  13 00068        BEQL    3$                          ;
                                                  7E  D4 0006A        CLRL    -(SP)                       ; 6451
                                              6B49  9F 0006C        PUSHAB  (R11)[SHLPTR]                   ;
                                                  9E  DD 0006F        PUSHL   @(SP)+                      ;
                                            14  AE  9F 00071        PUSHAB  GBLNAME                         ;
```

```
                                00000000'  EF   DD 00074          PUSHL   RST$START_ADDR
                          D6BC  CF         04   FB 0007A          CALLS   #4, DBG$GST_TO_RST
                                           0E   DD 0007F          PUSHL   #14                                          ; 6456
                          F9F5  CF         01   FB 00081          CALLS   #1, GET_MEM
                                58         50   D0 00086          MOVL    R0, RSTPTR
                                      6849 9F   00089            PUSHAB  (R11)[SHLPTR]                                ; 6457
                            30  A8         9E   D0 0008C          MOVL    @(SP)+, 48(RSTPTR)
                            34  A8         6E   90 00090          MOVB    CHANNEL_NUMBER, 52(RSTPTR)                   ; 6458
                            28  A8         10   88 00094          BISB2   #16, 40(RSTPTR)                              ; 6459
                                57     0C  AE   9A 00098          MOVZBL  GBLNAME, LENGTH                              ; 6466
                                57         08   C0 0009C          ADDL2   #8, LENGTH
                                50     03  A7   9E 0009F          MOVAB   3(R7), R0                                    ; 6467
                       7E                50   C7 000A3          DIVL3   #4, R0, -(SP)
                 00000000G      00         01   FB 000A7          CALLS   #1, DBG$GET_MEMORY
                            04  AE         50   D0 000AE          MOVL    R0, DSTPTR
                04  BE                     57   83 000B2          SUBB3   #1, LENGTH, @DSTPTR                          ; 6468
                    50          04  AE     01   C1 000B7          ADDL3   #1, DSTPTR, R0                               ; 6469
                                60     BC  8F   90 000BC          MOVB    #-68, (R0)
                    50          04  AE     03   C1 000C0          ADDL3   #3, DSTPTR, R0                               ; 6470
                                60         0A   D0 000C5          MOVL    #10, (R0)
                    5A          04  AE     07   C1 000C8          ADDL3   #7, DSTPTR, NAMEPTR                          ; 6471
                                6A     0C  AE   90 000CD          MOVB    GBLNAME, (NAMEPTR)                           ; 6472
                                50         6A   9A 000D1          MOVZBL  (NAMEPTR), R0                                ; 6473
                01  AA          0D  AE     50   28 000D4          MOVC3   R0, GBLNAME+1, 1(NAMEPTR)
                            0C  A8     04  AE   D0 000DA          MOVL    DSTPTR, 12(RSTPTR)                           ; 6474
                            14  A8         01   90 000DF          MOVB    #1, 20(RSTPTR)                               ; 6475
                            29  A8         0A   90 000E3          MOVB    #10, 41(RSTPTR)                              ; 6476
                                58         DD   000E7            PUSHL   RSTPTR                                       ; 6477
                          D826  CF         01   FB 000E9          CALLS   #1, DBG$HASH_INSERT
                                           7E   D4 000EE          CLRL    -(SP)                                        ; 6483
                                00000000'  EF   9F 000F0          PUSHAB  SAT$START_ADDR
                                58         DD   000F6            PUSHL   RSTPTR                                       ;
                                01         DD   000F8            PUSHL   #1                                           ;
                                      30   A8   DD 000FA          PUSHL   48(RSTPTR)                                   ;
                           0000V  CF         05   FB 000FD          CALLS   #5, STATIC_ADDRESS                           ;
                                50 00000000'  EF   D0 00102          MOVL    PREV_RST_PTR, R0                             ; 6489
                             10  A0         58   D0 00109          MOVL    RSTPTR, T6(R0)
                 00000000'       EF         58   D0 0010D          MOVL    RSTPTR, PREV_RST_PTR                         ; 6490
                                           07   11 00114          BRB     8$                                           ; 6443
                                03     15  A8   E8 00116 7$:       BLBS    21(RSTPTR), 8$                               ; 6496
                                      FF37 31   00011A            BRW     5$                                           ;
                       7E       08  AE         1C   C1 0011D 8$:  ADDL3   #28, IAFPTR, -(SP)                          ; 6413
                       02           56         9E   F2 00122      AOBLSS  @(SP)+, I, 9$                                ;
                                           03   11 00126          BRB     10$                                          ;
                                      FEFD 31   00128 9$:         BRW     4$                                           ;
                            08  AE     08  BE   D0 0012B 10$:     MOVL    @IAFPTR, IAFPTR                              ; 6504
                                      FEE2 31   00130            BRW     1$                                           ; 6404
                                           04   00133          RET                                                  ; 6512
                                         0000   00134 11$:       .WORD   Save nothing                                 ; 6368
                                           7E   D4 00136          CLRL    -(SP)                                        ;
                                           5E   DD 00138          PUSHL   SP                                           ;
                                7E     04  AC   7D 0013A          MOVQ    4(AP), -(SP)                                 ;
                          FE7E  CF         03   FB 0013E          CALLS   #3, SHARED_IMAGE_NAMES_HANDLER               ;
                                           04   00143          RET                                                  ;

; Routine Size:  324 bytes,    Routine Base:  DBG$CODE + 2E28
```

```
: 6414          6513  1
```

```
;  6416          6514  1 ROUTINE STATIC_ADDRESS(STARTADDR, LENGTH, RSTPTR, SAT_CHAIN, MERGE_FLAG): NOVALUE =
;  6417          6515  1 !
;  6418          6516  1 ! FUNCTION:
;  6419          6517  1 !       This routine constructs a Static Address Table (SAT) entry and inserts
;  6420          6518  1 !       it in the proper place in a specified SAT chain.  It accepts an address
;  6421          6519  1 !       range as input and searches the specified Static Address Table chain
;  6422          6520  1 !       until it finds the proper place to insert an entry for the new address
;  6423          6521  1 !       range.  The insertion is always done so that the SAT chain remains sort-
;  6424          6522  1 !       ed by start address.  If the MERGE_FLAG parameter is set, SAT entries
;  6425          6523  1 !       along the way whose address ranges overlap or adjoin the new address
;  6426          6524  1 !       range are removed from the SAT chain and their address ranges merged
;  6427          6525  1 !       with the new address range.
;  6428          6526  1 !
;  6429          6527  1 !       This routine is called during RST initialization, when a SAT chain is
;  6430          6528  1 !       built for each module in the DST.  These SAT chains are then merged (not
;  6431          6529  1 !       by this routine) to produce the Program Static Address Table which says
;  6432          6530  1 !       in which modules a given address can be found.  (An address may be found
;  6433          6531  1 !       in several modules--Fortran COMMON blocks cause this situation.)
;  6434          6532  1 !
;  6435          6533  1 !       This routine is also called when the RST and SAT for an individual mod-
;  6436          6534  1 !       ule is built.  In this case, the SAT is of course still sorted by start
;  6437          6535  1 !       address.  However, if two SAT entries have the same start address, they
;  6438          6536  1 !       are always sorted so that the larger end address comes first.  This
;  6439          6537  1 !       ordering is required by routine DBG$STA_SETCONTEXT which needs it to
;  6440          6538  1 !       determine when there are nested routines which overlap the address range
;  6441          6539  1 !       of an outer routine.  The ordering ensures that a nested routine always
;  6442          6540  1 !       follows an outer routine in the Module SAT.
;  6443          6541  1 !
;  6444          6542  1 ! INPUTS:
;  6445          6543  1 !       STARTADDR - The start address of the address range to be inserted in the
;  6446          6544  1 !                   specified Static Address Table Chain.  If the Program Static
;  6447          6545  1 !                   Address Table is built from the Debug Module Table (DMT),
;  6448          6546  1 !                   STARTADDR contains the address SAT$START_ADDR, in which case
;  6449          6547  1 !                   GET_MEM is called to get SAT space instead of DBG$GET_MEMORY.
;  6450          6548  1 !
;  6451          6549  1 !       LENGTH  - The length in bytes of the STARTADDR address range.
;  6452          6550  1 !
;  6453          6551  1 !       RSTPTR  - The Run-Time Symbol Table (RST) pointer which specifies the
;  6454          6552  1 !                   symbol to which this address range belongs.  When the Program
;  6455          6553  1 !                   Static Address Table (which says which module covers which
;  6456          6554  1 !                   address range) is built, RSTPTR is zero.  Then GET_MEM is
;  6457          6555  1 !                   called to get SAT space instead of DBG$GET_MEMORY.
;  6458          6556  1 !
;  6459          6557  1 !       SAT_CHAIN - The address of a longword location which contains a pointer
;  6460          6558  1 !                   to the first entry on the Static Address Table chain being
;  6461          6559  1 !                   built.  If the chain is empty, this location contains a zero.
;  6462          6560  1 !
;  6463          6561  1 !       MERGE_FLAG - A flag which specifies whether SAT entries on the specified
;  6464          6562  1 !                   chain should be merged if their address ranges adjoin or over-
;  6465          6563  1 !                   lap.  The flag can have these two values:
;  6466          6564  1 !
;  6467          6565  1 !                         TRUE  -- Merge SAT entries with adjoining or
;  6468          6566  1 !                                  overlapping address ranges.
;  6469          6567  1 !                         FALSE -- Do not merge such SAT entries.
;  6470          6568  1 !
;  6471          6569  1 ! OUTPUTS:
;  6472          6570  1 !       SAT_CHAIN - A new SAT entry is added to the SAT_CHAIN chain.  Also, old
```

```
;  6473     6571  1 !                 entries whose address ranges can be merged with the new one
;  6474     6572  1 !                 are removed from the chain if MERGE_FLAG is set.  Either oper-
;  6475     6573  1 !                 ation can change the value at the SAT_CHAIN location.
;  6476     6574  1 !
;  6477     6575  1 !       No value is returned by STATIC_ADDRESS.
;  6478     6576  1 !
;  6479     6577  1
;  6480     6578  2     BEGIN
;  6481     6579  2
;  6482     6580  2     MAP
;  6483     6581  2         RSTPTR: REF RST$ENTRY;          ! Pointer to symbol's RST entry
;  6484     6582  2
;  6485     6583  2     LOCAL
;  6486     6584  2         ADDR1,                          ! The start address of the input address
;  6487     6585  2                                         !      range, possibly as expanded by
;  6488     6586  2                                         !      merging with existing SAT entries
;  6489     6587  2         ADDR2,                          ! The end address of the input address
;  6490     6588  2                                         !      range, possibly as expanded by
;  6491     6589  2                                         !      merging with existing SAT entries
;  6492     6590  2         OLDPTR: REF SAT$ENTRY,          ! Pointer to the SAT entry just before
;  6493     6591  2                                         !      the SATPTR entry in the SAT chain
;  6494     6592  2         SATPTR: REF SAT$ENTRY;          ! Pointer to the current SAT entry in
;  6495     6593  2                                         !      the SAT chain.
;  6496     6594  2
;  6497     6595  2
;  6498     6596  2
;  6499     6597  2     ! Compute the start and end addresses of the address range passed by the
;  6500     6598  2     ! caller.  Also initialize the Static Address Table list pointers.
;  6501     6599  2     !
;  6502     6600  2     ADDR1 = .STARTADDR;
;  6503     6601  2     IF .LENGTH LEQ 0
;  6504     6602  2     THEN
;  6505     6603  2         ADDR2 = .STARTADDR
;  6506     6604  2     ELSE
;  6507     6605  2         ADDR2 = .STARTADDR + .LENGTH - 1;
;  6508     6606  2
;  6509     6607  2     OLDPTR = .SAT_CHAIN;
;  6510     6608  2     SATPTR = .OLDPTR[SAT$L_FLINK];
;  6511     6609  2
;  6512     6610  2
;  6513     6611  2     ! Loop through the entire Static Address Table linked list, looking to see
;  6514     6612  2     ! where we should link in an entry for this static address range or merge
;  6515     6613  2     ! it with existing SAT entries with overlapping address ranges.
;  6516     6614  2     !
;  6517     6615  2     WHILE .SATPTR NEQ 0 DO
;  6518     6616  3         BEGIN
;  6519     6617  3
;  6520     6618  3
;  6521     6619  3         ! If this SAT entry's address range comes before the ADDR1 start address
;  6522     6620  3         ! we just link on to the next SAT entry.
;  6523     6621  3         !
;  6524     6622  3         IF .SATPTR[SAT$L_END] + 1 LSSU .ADDR1
;  6525     6623  3         THEN
;  6526     6624  4             BEGIN
;  6527     6625  4             OLDPTR = .SATPTR;
;  6528     6626  4             SATPTR = .SATPTR[SAT$L_FLINK];
;  6529     6627  4             END
```

```
; 6530      6628  4
; 6531      6629  4
; 6532      6630  4          ! If this SAT entry's address range comes after the ADDR2 end address,
; 6533      6631  4          ! we have gone too far in the sorted SAT chain, so we exit the loop.
; 6534      6632  4          !
; 6535      6633  4          ELSE IF .ADDR2 + 1 LSSU .SATPTR[SAT$L_START]
; 6536      6634  4          THEN
; 6537      6635  4              EXITLOOP
; 6538      6636  4
; 6539      6637  4
; 6540      6638  4          ! The current SAT entry's address range overlaps or adjoins the ADDR1 -
; 6541      6639  4          ! ADDR2 address range.  If merging of such address ranges is requested,
; 6542      6640  4          ! we compute the expanded address range (including both ranges) in ADDR1
; 6543      6641  4          ! and ADDR2 and delete the SAT entry from the SAT chain.  We then keep
; 6544      6642  4          ! searching the chain in case other SAT entries overlap the new range.
; 6545      6643  4          !
; 6546      6644  4          ELSE IF .MERGE_FLAG
; 6547      6645  4          THEN
; 6548      6646  4              BEGIN
; 6549      6647  4              ADDR1 = MIN(.ADDR1, .SATPTR[SAT$L_START]);
; 6550      6648  4              ADDR2 = MAX(.ADDR2, .SATPTR[SAT$L_END]);
; 6551      6649  4              OLDPTR[SAT$L_FLINK] = .SATPTR[SAT$L_FLINK];
; 6552      6650  4              DBG$REL_MEMORY(.SATPTR);
; 6553      6651  4              SATPTR = .OLDPTR[SAT$L_FLINK];
; 6554      6652  4              END
; 6555      6653  4
; 6556      6654  4
; 6557      6655  4          ! The current SAT entry's address range overlaps the ADDR1 - ADDR2
; 6558      6656  4          ! address range, but merging address ranges is not allowed.  Exit the
; 6559      6657  4          ! loop if the ADDR1 start address is less than that in the SAT entry.
; 6560      6658  4          ! Also exit the loop if the ADDR1 start address equals that in the SAT
; 6561      6659  4          ! entry and the ADDR2 end address is greater than or equal to the SAT
; 6562      6660  4          ! entry's end address.  Otherwise, link to the next SAT entry.  This
; 6563      6661  4          ! ensures that the SAT chain will remain sorted on start address with
; 6564      6662  4          ! the larger end address first if two start addresses are the same.
; 6565      6663  4          !
; 6566      6664  3          ELSE IF (.ADDR1 LSSU .SATPTR[SAT$L_START]) OR
; 6567      6665  4                  (.ADDR1 EQLU .SATPTR[SAT$L_START] AND
; 6568      6666  4                   .ADDR2 GEQU .SATPTR[SAT$L_END])
; 6569      6667  3          THEN
; 6570      6668  3              EXITLOOP
; 6571      6669  3
; 6572      6670  3          ELSE
; 6573      6671  4              BEGIN
; 6574      6672  4              OLDPTR = .SATPTR;
; 6575      6673  4              SATPTR = .SATPTR[SAT$L_FLINK];
; 6576      6674  3              END;
; 6577      6675  3
; 6578      6676  2          END;                                ! End of WHILE loop
; 6579      6677  2
; 6580      6678  2
; 6581      6679  2      ! Build a new SAT entry containing the ADDR1 - ADDR2 address range and link
; 6582      6680  2      ! it into the SAT chain after the OLDPTR position.  Then return.
; 6583      6681  2      !
; 6584      6682  2      IF (.RSTPTR EQL 0) OR (.SAT_CHAIN EQL SAT$START_ADDR)
; 6585      6683  2      THEN
; 6586      6684  2          SATPTR = GET_MEM(SAT$K_ENTSIZE)
```

```
;  6587          6685  2
;  6588          6686  2      ELSE
;  6589          6687  2          SATPTR = DBG$GET_MEMORY(SAT$K_ENTSIZE);
;  6590          6688  2
;  6591          6689  2      IF .SATPTR EQL 0 THEN RETURN;
;  6592          6690  2      SATPTR[SAT$L_FLINK] = .OLDPTR[SAT$L_FLINK];
;  6593          6691  2      SATPTR[SAT$L_START] = .ADDR1;
;  6594          6692  2      SATPTR[SAT$L_END] = .ADDR2;
;  6595          6693  2      SATPTR[SAT$L_RSTPTR] = .RSTPTR;
;  6596          6694  2      OLDPTR[SAT$L_FLINK] = .SATPTR;
;  6597          6695  2
;  6598          6696  2
;  6599          6697  2      ! If RSTPTR points to a Routine RST Entry, put a pointer in that RST entry
;  6600          6698  2      ! to the SAT entry we just built.  This pointer is later used by routine
;  6601          6699  2      ! DBG$STA_SETCONTEXT to find a routine's SAT entry.
;  6602          6700  2      !
;  6603          6701  2      IF .RSTPTR NEQ 0
;  6604          6702  2      THEN
;  6605          6703  3          BEGIN
;  6606          6704  3          IF .RSTPTR[RST$B_KIND] EQL RST$K_ROUTINE
;  6607          6705  3          THEN
;  6608          6706  3              RSTPTR[RST$L_RTNSATPTR] = .SATPTR;
;  6609          6707  3
;  6610          6708  2          END;
;  6611          6709  2
;  6612          6710  2      RETURN;
;  6613          6711  2
;  6614          6712  1      END;

                                007C 00000 STATIC_ADDRESS:
                                           .WORD   Save R2,R3,R4,R5,R6                    ;  6514
                    53      04  AC  D0 00002       MOVL    STARTADDR, ADDR1               ;  6600
                            08  AC  D5 00006       TSTL    LENGTH                         ;  6601
                                06  14 00009       BGTR    1$                             ;
                    52      04  AC  D0 0000B       MOVL    STARTADDR, ADDR2               ;  6603
                                0A  11 0000F       BRB     2$                             ;
50          04      AC      08  AC  C1 00011 1$:    ADDL3   LENGTH, STARTADDR, R0          ;  6605
                    52      FF  A0  9E 00017       MOVAB   -1(R0), ADDR2                  ;
                    56      10  AC  D0 0001B 2$:    MOVL    SAT_CHAIN, OLDPTR              ;  6607
                    54          66  D0 0001F 3$:    MOVL    (OLDPTR), SATPTR               ;  6608
                                5C  13 00022 4$:    BEQL    9$                             ;  6615
50          08      A4          01  C1 00024       ADDL3   #1, 8(SATPTR), R0              ;  6622
                    53          50  D1 00029       CMPL    R0, ADDR1                      ;
                                4A  1F 0002C       BLSSU   8$                             ;
                    50      01  A2  9E 0002E       MOVAB   1(R2), R0                      ;  6633
                04          A4  50  D1 00032       CMPL    R0, 4(SATPTR)                  ;
                                48  1F 00036       BLSSU   9$                             ;
                    2E      14  AC  E9 00038       BLBC    MERGE_FLAG, 7$                 ;  6644
                    50          53  D0 0003C       MOVL    ADDR1, R0                      ;  6647
                04          A4  50  D1 0003F       CMPL    R0, 4(SATPTR)                  ;
                                04  15 00043       BLEQ    5$                             ;
                    50      04  A4  D0 00045       MOVL    4(SATPTR), R0                  ;
                    53          50  D0 00049 5$:    MOVL    R0, ADDR1                      ;
```

```
                50              52  D0 0004C          MOVL    ADDR2, R0                       ; 6648
          08    A4              50  D1 0004F          CMPL    R0, 8(SATPTR)                   ;
                                04  18 00053          BGEQ    6$                              ;
                50        08    A4  D0 00055          MOVL    8(SATPTR), R0                   ;
                52              50  D0 00059 6$:      MOVL    R0, ADDR2                       ;
                66              64  D0 0005C          MOVL    (SATPTR), (OLDPTR)              ; 6649
                                54  DD 0005F          PUSHL   SATPTR                          ; 6650
00000000G       00              01  FB 00061          CALLS   #1, DBG$REL_MEMORY              ;
                                B5  11 00068          BRB     3$                              ; 6651
          04    A4              53  D1 0006A 7$:      CMPL    ADDR1, 4(SATPTR)                ; 6664
                                10  1F 0006E          BLSSU   9$                              ;
                                06  12 00070          BNEQ    8$                              ; 6665
          08    A4              52  D1 00072          CMPL    ADDR2, 8(SATPTR)                ; 6666
                                08  1E 00076          BGEQU   9$                              ;
                56              54  D0 00078 8$:      MOVL    SATPTR, OLDPTR                  ; 6672
                54              64  D0 0007B          MOVL    (SATPTR), SATPTR                ; 6673
                                A2  11 0007E          BRB     4$                              ; 6615
                55        0C    AC  D0 00080 9$:      MOVL    RSTPTR, R5                      ; 6682
                                0D  13 00084          BEQL    10$                             ;
                50 00000000'    EF  9E 00086          MOVAB   SAT$START_ADDR, R0              ;
                50        10    AC  D1 0008D          CMPL    SAT_CHAIN, R0                   ;
                                09  12 00091          BNEQ    11$                             ;
                                05  DD 00093 10$:     PUSHL   #5                              ; 6684
          F89D  CF              01  FB 00095          CALLS   #1, GET_MEM                     ;
                                09  11 0009A          BRB     12$                             ;
                                05  DD 0009C 11$:     PUSHL   #5                              ; 6687
00000000G       00              01  FB 0009E          CALLS   #1, DBG$GET_MEMORY              ;
                54              50  D0 000A5 12$:     MOVL    R0, SATPTR                      ;
                                20  13 000AB          BEQL    13$                             ; 6689
                64              66  D0 000AA          MOVL    (OLDPTR), (SATPTR)              ; 6690
          04    A4              53  D0 000AD          MOVL    ADDR1, 4(SATPTR)                ; 6691
          08    A4              52  D0 000B1          MOVL    ADDR2, 8(SATPTR)                ; 6692
          0C    A4              55  D0 000B5          MOVL    R5, 12(SATPTR)                  ; 6693
                66              54  D0 000B9          MOVL    SATPTR, (OLDPTR)                ; 6694
                                55  D5 000BC          TSTL    R5                              ; 6701
                                0A  13 000BE          BEQL    13$                             ;
                02        14    A5  91 000C0          CMPB    20(R5), #2                      ; 6704
                                04  12 000C4          BNEQ    13$                             ;
          20    A5              54  D0 000C6          MOVL    SATPTR, 32(R5)                  ; 6706
                                    04 000CA 13$:     RET                                     ; 6712

; Routine Size:  203 bytes,    Routine Base:  DBG$CODE + 2F6C
```

```
;  6616          6713  1 ROUTINE STATIC_CHECK(DSTPTR, ADDRESS, LENGTH, STATIC_FLAG): NOVALUE =
;  6617          6714  1 !
;  6618          6715  1 ! FUNCTION
;  6619          6716  1 !       This routine determines whether a specified data item is static or not.
;  6620          6717  1 !       This is done by inspecting the data item's DST entry; the data object is
;  6621          6718  1 !       deemed to be static if the object's data address is specified directly
;  6622          6719  1 !       in the DST entry or if it is specified in a descriptor directly in the
;  6623          6720  1 !       DST entry.  Any use of registers or indirection in the address calcula-
;  6624          6721  1 !       tion disqualifies the data item from being deemed static.
;  6625          6722  1 !
;  6626          6723  1 !       If the data item is static, STATIC_FLAG is set to TRUE and the item's
;  6627          6724  1 !       start address and byte length is returned to the caller.  If the data
;  6628          6725  1 !       item is not static or if the DST entry is not even that of a data item,
;  6629          6726  1 !       STATIC_FLAG is set to FALSE and nothing else is returned.
;  6630          6727  1 !
;  6631          6728  1 ! INPUTS
;  6632          6729  1 !       DSTPTR  - A pointer to the Debug Symbol Table (DST) entry of the data
;  6633          6730  1 !                 item in question.
;  6634          6731  1 !
;  6635          6732  1 !       ADDRESS - The address of a longword location where the address of the
;  6636          6733  1 !                 static area is to be returned.
;  6637          6734  1 !
;  6638          6735  1 !       LENGTH  - The address of a longword location where the length of the
;  6639          6736  1 !                 static area is to be returned.
;  6640          6737  1 !
;  6641          6738  1 !       STATIC_FLAG - The address of a longword location where a flag should
;  6642          6739  1 !                 be returned saying whether the data item is static or not.
;  6643          6740  1 !
;  6644          6741  1 ! OUTPUTS
;  6645          6742  1 !       ADDRESS - If the data item is static, the start address of its static
;  6646          6743  1 !                 address range is returned to ADDRESS.
;  6647          6744  1 !
;  6648          6745  1 !       LENGTH  - If the data item is static, the length in bytes of its static
;  6649          6746  1 !                 address range is returned to LENGTH.
;  6650          6747  1 !
;  6651          6748  1 !       STATIC_FLAG - If the data item is static, STATIC_FLAG is set to TRUE;
;  6652          6749  1 !                 otherwise STATIC_FLAG is set to FALSE.
;  6653          6750  1 !
;  6654          6751  1
;  6655          6752  2     BEGIN
;  6656          6753  2
;  6657          6754  2     ENABLE
;  6658          6755  2         STATIC_CHECK_HANDLER;
;  6659          6756  2
;  6660          6757  2     MAP
;  6661          6758  2         DSTPTR: REF DST$RECORD,          ! Pointer to data item's DST record
;  6662          6759  2         ADDRESS: REF VECTOR[1],          ! Location where the static start
;  6663          6760  2                                          !      address is returned
;  6664          6761  2         LENGTH: REF VECTOR[1],           ! Location where the length of the
;  6665          6762  2                                          !      static area is returned
;  6666          6763  2         STATIC_FLAG: REF VECTOR[1];      ! Location where TRUE is returned if the
;  6667          6764  2                                          !      data item is static--else FALSE
;  6668          6765  2
;  6669          6766  2     BIND
;  6670          6767  2         LENGTH_TBL = UPLIT BYTE(0, 1, 1,2,4,8, 1,2,4,8, 4,8,8,16,
;  6671          6768  2                                 REP 11 OF (0), 16,16,8,16,16,32,
;  6672          6769  2                                 12,8,8,0,0,0,0): VECTOR[,BYTE];
```

```
;  6673        L 6770  2          %IF DSC$K_DTYPE_HIGHEST GTR 37  ! If the LENGTH_TBL PLIT is too small,
;  6674        U 6771  2          %THEN                          !     generate a compile-time error
;  6675        U 6772  2              %ERROR('Error:  Must expand PLIT in Routine STATIC_CHECK')
;  6676          6773  2          %FI;
;  6677          6774  2
;  6678          6775  2      LOCAL
;  6679          6776  2          BPTR1: REF DST$BLI_TRAILER1,    ! Pointer to the Bliss special cases DST
;  6680          6777  2                                         !     record first trailer
;  6681          6778  2          BPTR2: REF DST$BLI_TRAILER2,    ! Pointer to the Bliss special cases DST
;  6682          6779  2                                         !     record second trailer
;  6683          6780  2          DESCPTR: REF BLOCK[,BYTE];      ! Pointer to VAX Standard Descriptor in
;  6684          6781  2                                         !     the DST record
;  6685          6782  2
;  6686          6783  2
;  6687          6784  2      ! Assume everything is not static to begin with.
;  6688          6785  2      !
;  6689          6786  2      STATIC_FLAG[0] = FALSE;
;  6690          6787  2
;  6691          6788  2
;  6692          6789  2      ! If this is a BLISS special cases DST record, return its start address and
;  6693          6790  2      ! length if the BLISS data item is static.
;  6694          6791  2      !
;  6695          6792  2      IF .DSTPTR[DST$B_TYPE] EQL DST$K_BLI
;  6696          6793  2      THEN
;  6697          6794  3          BEGIN
;  6698          6795  3          STATIC_FLAG[0] = FALSE;
;  6699          6796  3          IF .DSTPTR[DST$B_BLI_VFLAGS] NEQ DST$K_VALKIND_ADDR THEN RETURN;
;  6700          6797  3          BPTR1 = DSTPTR[DST$A_BLI_TRLR1] + .DSTPTR[DST$B_BLI_LNG];
;  6701          6798  3          ADDRESS[0] = .BPTR1[DST$L_BLI_VALUE];
;  6702          6799  3          BPTR2 = BPTR1[DST$A_BLI_TRLR2] + .BPTR1[DST$B_BLI_NAME];
;  6703          6800  3          LENGTH[0] = .BPTR2[DST$L_BLI_SIZE];
;  6704          6801  3          IF .DSTPTR[DST$V_BLI_REF] THEN LENGTH[0] = 4;
;  6705          6802  3          STATIC_FLAG[0] = TRUE;
;  6706          6803  3          RETURN;
;  6707          6804  2          END;
;  6708          6805  2
;  6709          6806  2
;  6710          6807  2      ! If this is a boolean (as in PASCAL) and it is static, we mark it as static
;  6711          6808  2      ! and return its address and a length of 1 byte.  We determine whether it is
;  6712          6809  2      ! static by checking the Value-flags field.
;  6713          6810  2      !
;  6714          6811  2      IF (.DSTPTR[DST$B_TYPE] EQL DST$K_BOOL) AND
;  6715          6812  3         (.DSTPTR[DST$B_VFLAGS] EQL DST$K_VALKIND_ADDR)
;  6716          6813  2      THEN
;  6717          6814  3          BEGIN
;  6718          6815  3          ADDRESS[0] = .DSTPTR[DST$L_VALUE];
;  6719          6816  3          LENGTH[0] = 1;
;  6720          6817  3          STATIC_FLAG[0] = TRUE;
;  6721          6818  3          RETURN;
;  6722          6819  2          END;
;  6723          6820  2
;  6724          6821  2
;  6725          6822  2      ! If this is a record begin record, or if the item is specified by the
;  6726          6823  2      ! DST$K_SEPTYP record, and it is static, then we mark it as static and
;  6727          6824  2      ! return its address.  However, we do not know its length (at least not
;  6728          6825  2      ! readily), so we return a length of zero.  In effect, we treat the
;  6729          6826  2      ! object as a label.
```

```
;6730      6827  2      !
;6731      6828  2      IF (.DSTPTR[DST$B_TYPE] EQL DST$K_SEPTYP) AND
;6732      6829  3         (.DSTPTR[DST$B_VFLAGS] EQL DST$K_VALKIND_ADDR)
;6733      6830  2      THEN
;6734      6831  3          BEGIN
;6735      6832  3          LOCAL
;6736      6833  3              TYPEPTR, TYPESPEC;
;6737      6834  3
;6738      6835  3          ADDRESS[0] = .DSTPTR[DST$L_VALUE];
;6739      6836  3          LENGTH[0] = DBG$FIND_SEPTYP(.DSTPTR, TYPEPTR, TYPESPEC);
;6740      6837  3          LENGTH[0] = (.LENGTH[0] + 7) / 8;
;6741      6838  3          STATIC_FLAG[0] = TRUE;
;6742      6839  3          RETURN;
;6743      6840  2          END;
;6744      6841  2
;6745      6842  2
;6746      6843  2      IF (.DSTPTR[DST$B_TYPE] EQL DST$K_RECBEG) AND
;6747      6844  3         (.DSTPTR[DST$B_VFLAGS] EQL DST$K_VALKIND_ADDR)
;6748      6845  2      THEN
;6749      6846  3          BEGIN
;6750      6847  3          LOCAL
;6751      6848  3              RECBEG_TRLR: REF DST$RECBEG_TRLR;
;6752      6849  3
;6753      6850  3          ADDRESS[0] = .DSTPTR[DST$L_VALUE];
;6754      6851  3          RECBEG_TRLR = DSTPTR[DST$B_NAME] + 1 + .DSTPTR[DST$B_NAME];
;6755      6852  3          LENGTH[0] = (.RECBEG_TRLR[DST$L_RECBEG_SIZE] + 7) / 8;
;6756      6853  3          STATIC_FLAG[0] = TRUE;
;6757      6854  3          RETURN;
;6758      6855  2          END;
;6759      6856  2
;6760      6857  2      IF (.DSTPTR[DST$B_TYPE] EQL DST$K_ENUMBEG) AND
;6761      6858  3         (.DSTPTR[DST$B_VFLAGS] EQL DST$K_VALKIND_ADDR)
;6762      6859  2      THEN
;6763      6860  3          BEGIN
;6764      6861  3          ADDRESS[0] = .DSTPTR[DST$L_VALUE];
;6765      6862  3          LENGTH[0] = (.DSTPTR[DST$B_ENUMBEG_LENG] + 7) / 8;
;6766      6863  3          STATIC_FLAG[0] = TRUE;
;6767      6864  2          END;
;6768      6865  2
;6769      6866  2
;6770      6867  2      ! If the DST record is anything other than a VAX Standard Data Type at this
;6771      6868  2      ! point, just return with STATIC_FLAG set to FALSE.
;6772      6869  2      !
;6773      6870  2      IF (.DSTPTR[DST$B_TYPE] LSS DSC$K_DTYPE_LOWEST) OR
;6774      6871  3         (.DSTPTR[DST$B_TYPE] GTR DSC$K_DTYPE_HIGHEST)
;6775      6872  2      THEN
;6776      6873  3          BEGIN
;6777      6874  3          STATIC_FLAG[0] = FALSE;
;6778      6875  3          RETURN;
;6779      6876  2          END;
;6780      6877  2
;6781      6878  2
;6782      6879  2      ! We have a VAX Standard data item.  If this is a static scalar, pick up its
;6783      6880  2      ! start address and length and return with STATIC_FLAG set to TRUE.
;6784      6881  2      !
;6785      6882  2      IF .DSTPTR[DST$B_VFLAGS] EQL DST$K_VALKIND_ADDR
;6786      6883  2      THEN
```

```
;  6787          6884  3          BEGIN
;  6788          6885  3          ADDRESS[0] = .DSTPTR[DST$L_VALUE];
;  6789          6886  3          LENGTH[0] = .LENGTH_TBL[.DSTPTR[DST$B_TYPE]];
;  6790          6887  3          STATIC_FLAG[0] = TRUE;
;  6791          6888  3          RETURN;
;  6792          6889  2          END;
;  6793          6890  2
;  6794          6891  2
;  6795          6892  2      ! If this data item DST record contains a VAX Standard Descriptor, we can
;  6796          6893  2      ! assume it to be static and we pick up its address and length from the
;  6797          6894  2      ! embedded descriptor.
;  6798          6895  2      !
;  6799          6896  2      IF .DSTPTR[DST$B_VFLAGS] EQL DST$K_VFLAGS_DSC
;  6800          6897  2      THEN
;  6801          6898  3          BEGIN
;  6802          6899  3
;  6803          6900  3
;  6804          6901  3          ! Get the start address and data item length from the descriptor.
;  6805          6902  3          !
;  6806          6903  3          DESCPTR = DSTPTR[DST$A_DSC_BASE] + .DSTPTR[DST$L_DSC_OFFS];
;  6807          6904  3          ADDRESS[0] = .DESCPTR[DSC$A_POINTER];
;  6808          6905  3          LENGTH[0] = DBG$GET_BITSIZE_FROM_DESC(.DESCPTR);
;  6809          6906  3          LENGTH[0] = (.LENGTH[0] + 7) / 8;
;  6810          6907  3
;  6811          6908  3          ! Set the static flag to TRUE and return.
;  6812          6909  3          !
;  6813          6910  3          STATIC_FLAG[0] = TRUE;
;  6814          6911  3          RETURN;
;  6815          6912  2          END;
;  6816          6913  2
;  6817          6914  2
;  6818          6915  2      ! If we have come here, the data item is not static.  Return saying so.
;  6819          6916  2      !
;  6820          6917  2      STATIC_FLAG[0] = FALSE;
;  6821          6918  2      RETURN;
;  6822          6919  2
;  6823          6920  1      END;


                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

    10  08  08  04  08  04  02  01  08  04  02  01  01  00  001E0 P.AAZ:  .BYTE   0, 1, 1, 2, 4, 8, 1, 2, 4, 8, 4, 8, 8, 16   ;
                                                        00# 001EE        .BYTE   0[11]                                      ;
        00  00  00  00  08  0E  0C  20  10  10  08  10  10  001F9        .BYTE   16, 16, 8, 16, 16, 32, 12, 8, 8, 0, 0, 0, -;
                                                                         0                                                  ;

                                                LENGTH_TBL=         P.AAZ


                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                001C 00000 STATIC_CHECK:
                                                        .WORD   Save R2,R3,R4                                       ; 6713
                                    5E          08  C2 00002        SUBL2   #8, SP                                      ;
                                    6D    010E  CF  DE 00005        MOVAL   13$, (FP)                                   ; 6752
                                    54      10  AC  D0 0000A        MOVL    STATIC_FLAG, R4                             ; 6786
```

```
                                64  D4 0000E          CLRL    (R4)
                            52  04  AC  D0 00010      MOVL    DSTPTR, R2                  ; 6792
                            53  01  A2  9A 00014      MOVZBL  1(R2), R3
                                2E  12 00018          BNEQ    2$
                                64  D4 0001A          CLRL    (R4)                        ; 6795
                            01  04  A2  91 0001C      CMPB    4(R2), #1                   ; 6796
                                01  13 00020          BEQL    1$
                                    04 00022          RET
                            50  02  A2  9A 00023 1$:   MOVZBL  2(R2), R0                  ; 6797
                            50  03 A042  9E 00027      MOVAB   3(R0)[R2], BPTR1
                        08  BC      60  D0 0002C      MOVL    (BPTR1), @ADDRESS           ; 6798
                            51  04  A0  9A 00030      MOVZBL  4(BPTR1), R1                ; 6799
                            50  05 A140  9E 00034      MOVAB   5(R1)[BPTR1], BPTR2
                        0C  BC      60  D0 00039      MOVL    (BPTR2), @LENGTH            ; 6800
                            05  A2  95 0003D          TSTB    5(R2)                       ; 6801
                                1B  18 00040          BGEQ    3$
                        0C  BC      04  D0 00042      MOVL    #4, @LENGTH
                                15  11 00046          BRB     3$                          ; 6802
                        9E  8F      53  91 00048 2$:   CMPB    R3, #158                    ; 6811
                                12  12 0004C          BNEQ    4$
                            01  02  A2  91 0004E      CMPB    2(R2), #1                   ; 6812
                                0C  12 00052          BNEQ    4$
                        08  BC  03  A2  D0 00054      MOVL    3(R2), @ADDRESS             ; 6815
                        0C  BC      01  D0 00059      MOVL    #1, @LENGTH                 ; 6816
                                00B0 31 0005D 3$:     BRW     11$                         ; 6817
                        A3  8F      53  91 00060 4$:   CMPB    R3, #163                    ; 6828
                                1C  12 00064          BNEQ    5$
                            01  02  A2  91 00066      CMPB    2(R2), #1                   ; 6829
                                16  12 0006A          BNEQ    5$
                        08  BC  03  A2  D0 0006C      MOVL    3(R2), @ADDRESS             ; 6835
                                5E  DD 00071          PUSHL   SP                          ; 6836
                                08  AE  9F 00073      PUSHAB  TYPEPTR
                                52  DD 00076          PUSHL   R2
                    00000000G   00      03  FB 00078 CALLS   #3, DBG$FIND_SEPTYP
                                0080 31 0007F         BRW     9$
                            51  01  A2  9A 00082 5$:   MOVZBL  1(R2), R1                  ; 6843
                        AB  8F      51  91 00086      CMPB    R1, #171
                                1A  12 0008A          BNEQ    6$
                            01  02  A2  91 0008C      CMPB    2(R2), #1                   ; 6844
                                14  12 00090          BNEQ    6$
                        08  BC  03  A2  D0 00092      MOVL    3(R2), @ADDRESS             ; 6850
                            50  07  A2  9A 00097      MOVZBL  7(R2), R0                   ; 6851
                            50  08 A042  9E 0009B      MOVAB   8(R0)[R2], RECBEG_TRLR
                50              60      07  C1 000A0  ADDL3   #7, (RECBEG_TRLR), R0       ; 6852
                                65  11 000A4          BRB     10$
                        A5  8F      51  91 000A6 6$:   CMPB    R1, #165                    ; 6857
                                1A  12 000AA          BNEQ    7$
                            01  02  A2  91 000AC      CMPB    2(R2), #1                   ; 6858
                                14  12 000B0          BNEQ    7$
                        08  BC  03  A2  D0 000B2      MOVL    3(R2), @ADDRESS             ; 6861
                            50  02  A2  9A 000B7      MOVZBL  2(R2), R0                   ; 6862
                            50      07  C0 000BB      ADDL2   #7, R0
                    0C  BC          50      08  C7 000BE  DIVL3   #8, R0, @LENGTH
                            64          01  D0 000C3  MOVL    #1, (R4)                    ; 6863
                                        51  D5 000C6 7$:   TSTL    R1                      ; 6870
                                        4A  15 000C8      BLEQ    12$
                            25          51  91 000CA      CMPB    R1, #37                 ; 6871
```

```
                                45  1A 000CD          BGTRU   12$                                                     ;
                        01      02  A2  91 000CF          CMPB    2(R2), #1                                           ; 6882
                                10  12 000D3          BNEQ    8$                                                      ;
                    08  BC      03  A2  D0 000D5          MOVL    3(R2), @ADDRESS                                     ; 6885
                    0C  BC 00000000'EF  41  9A 000DA      MOVZBL  LENGTH_TBL[R1], @LENGTH                             ; 6886
                                2B  11 000E3          BRB     11$                                                     ; 6887
                    FA  8F      02  A2  91 000E5  8$:     CMPB    2(R2), #250                                         ; 6896
                                28  12 000EA          BNEQ    12$                                                     ;
                        52      03  A2  C0 000EC          ADDL2   3(R2), R2                                           ; 6903
                        50      07  A2  9E 000F0          MOVAB   7(R2), DESCPTR                                      ;
                    08  BC      04  A0  D0 000F4          MOVL    4(DESCPTR), @ADDRESS                                ; 6904
                                50  DD 000F9          PUSHL   DESCPTR                                                 ; 6905
                00000000G   00      01  FB 000FB          CALLS   #1, DBG$GET_BITSIZE_FROM_DESC                       ;
                    0C  BC          50  D0 00102  9$:     MOVL    R0, @LENGTH                                         ;
            50      0C  BC          07  C1 00106          ADDL3   #7, @LENGTH, R0                                     ; 6906
        0C  BC              50      08  C7 0010B  10$:     DIVL3   #8, R0, @LENGTH                                     ;
                        64      01  D0 00110  11$:     MOVL    #1, (R4)                                            ; 6910
                                    04 00113          RET                                                             ; 6898
                                64  D4 00114  12$:     CLRL    (R4)                                                ; 6917
                                    04 00116          RET                                                             ; 6920
                                  0000 00117  13$:     .WORD   Save nothing                                        ; 6752
                                7E  D4 00119          CLRL    -(SP)                                                   ;
                                5E  DD 0011B          PUSHL   SP                                                      ;
                        7E      04  AC  7D 0011D          MOVQ    4(AP), -(SP)                                        ;
                0000V   CF          03  FB 00121          CALLS   #3, STATIC_CHECK_HANDLER                            ;
                                    04 00126          RET                                                             ;

; Routine Size:  295 bytes,    Routine Base:  DBG$CODE + 3037
```

```
;  6825          6921  1 ROUTINE STATIC_CHECK_HANDLER(SIGARG, MECHARG, ENBLARG) =
;  6826          6922  1 !
;  6827          6923  1 ! FUNCTION
;  6828          6924  1 !       This routine is to catch the signals for STATIC_CHECK routine.
;  6829          6925  1 !       So the STATIC_CHECK will not treat the object to be static.
;  6830          6926  1 !
;  6831          6927  1 ! INPUTS
;  6832          6928  1 !       SIGARG  - The signal argument vector.
;  6833          6929  1 !
;  6834          6930  1 !       MECHARG - The mechanism argument vector.
;  6835          6931  1 !
;  6836          6932  1 !       ENBLARG - The enable argument vector. (not used)
;  6837          6933  1 !
;  6838          6934  1 ! OUTPUTS
;  6839          6935  1 !       None.
;  6840          6936  1 !
;  6841          6937  1
;  6842          6938  2     BEGIN
;  6843          6939  2
;  6844          6940  2     MAP
;  6845          6941  2         SIGARG: REF VECTOR[,LONG];
;  6846          6942  2
;  6847          6943  2     IF .SIGARG[1] EQL DBG$_ACCADDCOM OR
;  6848          6944  2        .SIGARG[1] EQL DBG$_DESCNOTSET OR
;  6849          6945  2        .SIGARG[1] EQL DBG$_SYMNOTACT OR
;  6850          6946  2        .SIGARG[1] EQL DBG$_NOACCESSR
;  6851          6947  2     THEN
;  6852          6948  3         BEGIN
;  6853          6949  3         SETUNWIND();
;  6854          6950  3         RETURN 0;
;  6855          6951  2         END;
;  6856          6952  2
;  6857          6953  2     RETURN SS$_RESIGNAL;
;  6858          6954  1     END;


                                0000 00000 STATIC_CHECK_HANDLER:
                                           .WORD   Save nothing                    ;  6921
                50          04  AC  D0 00002          MOVL    SIGARG, R0         ;  6943
00028C98        8F          04  A0  D1 00006          CMPL    4(R0), #167064
                                1E  13 0000E          BEQL    1$
00028F50        8F          04  A0  D1 00010          CMPL    4(R0), #167760     ;  6944
                                14  13 00018          BEQL    1$
00028C88        8F          04  A0  D1 0001A          CMPL    4(R0), #167048     ;  6945
                                0A  13 00022          BEQL    1$
00028228        8F          04  A0  D1 00024          CMPL    4(R0), #164392     ;  6946
                                0B  12 0002C          BNEQ    2$
                                7E  7C 0002E 1$:      CLRQ    -(SP)              ;  6949
00000000G       00              02  FB 00030          CALLS   #2, SYS$UNWIND
                                06  11 00037          BRB     3$                 ;  6950
                50      0918    8F  3C 00039 2$:      MOVZWL  #2328, R0          ;  6953
                                04  0003E            RET
                                50  D4 0003F 3$:      CLRL    R0                 ;  6954
                                04  00041            RET
```

; Routine Size: 66 bytes, Routine Base: DBG$CODE + 315E

; 6859 6955 1

```
;  6861          6956  1 ROUTINE TEST_DST_SIZE(FILENAME, IMG_DST_BLOCK) =
;  6862          6957  1 !
;  6863          6958  1 ! FUNCTION
;  6864          6959  1 !       This routine is used only if this is a Testable Debugger.  It scans the
;  6865          6960  1 !       Debug Symbol Table (DST) to find where the User Program DST ends.  This
;  6866          6961  1 !       is done to exclude the Debugger's own DST from the DST mapped into the
;  6867          6962  1 !       virtual address space.  The routine thus scans the DST (using RMS I/O so
;  6868          6963  1 !       that no virtual address space is used up) looking for the Module-Begin
;  6869          6964  1 !       record for DBGSTART.  If this record is found, that location is returned
;  6870          6965  1 !       as the DST end location; otherwise the actual end of the DST is returned
;  6871          6966  1 !       as the end location.
;  6872          6967  1 !
;  6873          6968  1 ! INPUTS
;  6874          6969  1 !       FILENAME - The address of a Counted ASCII string containing the file-
;  6875          6970  1 !                  name of the executable image file.
;  6876          6971  1 !
;  6877          6972  1 !       IMG_DST_BLOCK - A pointer to the image header DST/GST descriptor.  This
;  6878          6973  1 !                  is assumed to already be in memory.
;  6879          6974  1 !
;  6880          6975  1 ! OUTPUTS
;  6881          6976  1 !       The number of bytes in the DST before the end of the DST or before the
;  6882          6977  1 !                  Module-Begin record for DBGSTART is returned as the value.
;  6883          6978  1 !
;  6884          6979  1
;  6885          6980  2     BEGIN
;  6886          6981  2
;  6887          6982  2     MAP
;  6888          6983  2         FILENAME: REF VECTOR[,BYTE],      ! Pointer to the executable image file-
;  6889          6984  2                                           !      name in Counted ASCII
;  6890          6985  2         IMG_DST_BLOCK: REF IHS$ENTRY;     ! Pointer to the image header symbol
;  6891          6986  2                                           !      table descriptor
;  6892          6987  2
;  6893          6988  2     LITERAL
;  6894          6989  2         BLKCOUNT = 10,                    ! Number of file blocks/buffer "block"
;  6895          6990  2         BUFSIZE  = (BLKCOUNT + 1)*512;    ! Size of the DST record buffer
;  6896          6991  2
;  6897          6992  2     OWN
;  6898          6993  2         DSTFAB: $FAB(FAC=<BIO,GET>),      ! DST File Access Block (FAB)
;  6899          6994  2         DSTRAB: $RAB(ROP=BIO);            ! DST Record Access Block (RAB)
;  6900          6995  2
;  6901          6996  2     LOCAL
;  6902          6997  2         BLKNUM,                           ! Block number within the image file--a
;  6903          6998  2                                           !      "block" is BLKCOUNT*512 bytes
;  6904          6999  2         BYTNUM,                           ! Byte number within current "block"
;  6905          7000  2         CURBLK,                           ! Number of current "block" in XBUFFER
;  6906          7001  2         DSTADDR,                          ! Current byte address in the DST rela-
;  6907          7002  2                                           !      tive to the start of the DST
;  6908          7003  2         DSTENDADDR,                       ! Byte address of the end of the DST
;  6909          7004  2         DSTPTR: REF DST$RECORD,           ! Pointer to the current DST record
;  6910          7005  2         STATUS,                           ! Status code returned by RMS
;  6911          7006  2         XBUFFER: VECTOR[BUFSIZE,BYTE];    ! DST file buffer
;  6912          7007  2
;  6913          7008  2
;  6914          7009  2
;  6915          7010  2     ! Set up the FAB and the RAB for doing Block I/O to read the executable
;  6916          7011  2     ! image file.  Then open and connect the executable image file.
;  6917          7012  2     !
```

```
;  6918          7013  2     DSTFAB[FAB$L_FNA] = FILENAME[1];
;  6919          7014  2     DSTFAB[FAB$B_FNS] = .FILENAME[0];
;  6920          7015  2     DSTRAB[RAB$L_FAB] = DSTFAB;
;  6921          7016  2     DSTRAB[RAB$L_UBF] = XBUFFER;
;  6922          7017  2     STATUS = $OPEN(FAB=DSTFAB);
;  6923          7018  2     IF NOT .STATUS THEN SIGNAL(.STATUS);
;  6924          7019  2     STATUS = $CONNECT(RAB=DSTRAB);
;  6925          7020  2     IF NOT .STATUS THEN SIGNAL(.STATUS);
;  6926          7021  2
;  6927          7022  2
;  6928          7023  2     ! Now loop over the whole Debug Symbol Table (DST) to find where the user
;  6929          7024  2     ! program DST ends.  This is either at the true end of the DST or at the
;  6930          7025  2     ! start of the Module-Begin DST record for DBGSTART.
;  6931          7026  2     !
;  6932          7027  2     CURBLK = -1;
;  6933          7028  2     DSTADDR = 0;
;  6934          7029  2     DSTENDADDR = 512*.IMG_DST_BLOCK[IHS$W_DSTBLKS] - 1;
;  6935          7030  2     WHILE TRUE DO
;  6936          7031  3         BEGIN
;  6937          7032  3
;  6938          7033  3
;  6939          7034  3         ! Find the "block" number and the byte offset within the block repre-
;  6940          7035  3         ! sented by the current DST address.  If that "block" of the file is
;  6941          7036  3         ! not now in the buffer, read it into the buffer.
;  6942          7037  3         !
;  6943          7038  3         BLKNUM = .DSTADDR/(BLKCOUNT*512);
;  6944          7039  3         BYTNUM = .DSTADDR MOD (BLKCOUNT*512);
;  6945          7040  3         IF .BLKNUM NEQ .CURBLK
;  6946          7041  3         THEN
;  6947          7042  4             BEGIN
;  6948          7043  4             DSTRAB[RAB$L_BKT] = .BLKNUM*BLKCOUNT + .IMG_DST_BLOCK[IHS$L_DSTVBN];
;  6949          7044  4             DSTRAB[RAB$W_USZ] = MIN(BUFSIZE,
;  6950          7045  4                                     .DSTENDADDR + 1 - .BLKNUM*BLKCOUNT*512);
;  6951          7046  4             STATUS = $READ(RAB=DSTRAB);
;  6952          7047  4             IF NOT .STATUS THEN SIGNAL(.STATUS);
;  6953          7048  4             CURBLK = .BLKNUM;
;  6954          7049  3             END;
;  6955          7050  3
;  6956          7051  3         DSTPTR = XBUFFER[.BYTNUM];
;  6957          7052  3
;  6958          7053  3
;  6959          7054  3         ! Determine whether we have reached the end of the DST.  If we have,
;  6960          7055  3         ! return the current DST size in bytes to the caller.
;  6961          7056  3         !
;  6962          7057  3         IF (.DSTADDR GEQU .DSTENDADDR)   OR
;  6963          7058  3            (.DSTPTR[DST$B_LENGTH] EQL 0) OR
;  6964          7059  4            (.DSTPTR[DST$B_TYPE] EQL DST$K_MODBEG AND
;  6965          7060  4             CH$EQL(9, UPLIT BYTE(%ASCIC 'DBGSTART'),
;  6966          7061  4                 .DSTPTR[DST$B_MODBEG_NAME] + 1, DSTPTR[DST$B_MODBEG_NAME], 0))
;  6967          7062  3         THEN
;  6968          7063  3             RETURN .DSTADDR;
;  6969          7064  3
;  6970          7065  3
;  6971          7066  3         ! We are not yet at the end of the DST.  Go to the next record and loop.
;  6972          7067  3         !
;  6973          7068  3         DSTADDR = .DSTADDR + 1 + .DSTPTR[DST$B_LENGTH];
;  6974          7069  2         END;
```

```
; 6975          7070  2
; 6976          7071  2
; 6977          7072  2      ! We should never get here.
; 6978          7073  2      !
; 6979          7074  2      RETURN 0;
; 6980          7075  2
; 6981          7076  1      END;


                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                    54  52  41  54  53  47  42  44  08  00206 P.ABA:   .ASCII  <8>\DBGSTART\                    ;

                                                        .PSECT  DBG$OWN,NOEXE,  PIC,2

                                                  03    04EF0 DSTFAB:  .BYTE   3                              ;
                                                  50    04EF1          .BYTE   80                             ;
                                                0000    04EF2          .WORD   0                              ;
                                            00000000    04EF4          .LONG   0                              ;
                                            00000000    04EF8          .LONG   0                              ;
                                            00000000    04EFC          .LONG   0                              ;
                                            00000000    04F00          .LONG   0                              ;
                                                0000    04F04          .WORD   0                              ;
                                                  22    04F06          .BYTE   34                             ;
                                                  00    04F07          .BYTE   0                              ;
                                            00000000    04F08          .LONG   0                              ;
                                                  00    04F0C          .BYTE   0                              ;
                                                  00    04F0D          .BYTE   0                              ;
                                                  00    04F0E          .BYTE   0                              ;
                                                  02    04F0F          .BYTE   2                              ;
                                            00000000    04F10          .LONG   0                              ;
                                            00000000    04F14          .LONG   0                              ;
                                            00000000    04F18          .LONG   0                              ;
                                            00000000    04F1C          .LONG   0                              ;
                                            00000000    04F20          .LONG   0                              ;
                                                  00    04F24          .BYTE   0                              ;
                                                  00    04F25          .BYTE   0                              ;
                                                0000    04F26          .WORD   0                              ;
                                            00000000    04F28          .LONG   0                              ;
                                                0000    04F2C          .WORD   0                              ;
                                                  00    04F2E          .BYTE   0                              ;
                                                  00    04F2F          .BYTE   0                              ;
                                            00000000    04F30          .LONG   0                              ;
                                            00000000    04F34          .LONG   0                              ;
                                                0000    04F38          .WORD   0                              ;
                                                  00    04F3A          .BYTE   0                              ;
                                                  00    04F3B          .BYTE   0                              ;
                                            00000000    04F3C          .LONG   0                              ;
                                                  01    04F40 DSTRAB:  .BYTE   1                              ;
                                                  44    04F41          .BYTE   68                             ;
                                                0000    04F42          .WORD   0                              ;
                                            00000800    04F44          .LONG   2048                           ;
                                            00000000    04F48          .LONG   0                              ;
                                            00000000    04F4C          .LONG   0                              ;
                                                0000#   04F50          .WORD   0[3]                           ;
                                                0000    04F56          .WORD   0                              ;
```

```
                                00000000  04F58          .LONG   0                                       ;
                                    0000  04F5C          .WORD   0                                       ;
                                      00  04F5E          .BYTE   0                                       ;
                                      00  04F5F          .BYTE   0                                       ;
                                    0000  04F60          .WORD   0                                       ;
                                    0000  04F62          .WORD   0                                       ;
                                00000000  04F64          .LONG   0                                       ;
                                00000000  04F68          .LONG   0                                       ;
                                00000000  04F6C          .LONG   0                                       ;
                                00000000  04F70          .LONG   0                                       ;
                                      00  04F74          .BYTE   0                                       ;
                                      00  04F75          .BYTE   0                                       ;
                                      00  04F76          .BYTE   0                                       ;
                                      00  04F77          .BYTE   0                                       ;
                                00000000  04F78          .LONG   0                                       ;
                                00000000  04F7C          .LONG   0                                       ;
                                00000000  04F80          .LONG   0                                       ;

                                                         .EXTRN  SYS$OPEN, SYS$CONNECT
                                                         .EXTRN  SYS$READ

                                                         .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                    0FFC 00000 TEST_DST_SIZE:
                                                         .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11    ; 6956
                                5E      EA00    CE   9E 00002    MOVAB   -5632(SP), SP                   ;
00000000'  EF       04          AC           01   C1 00007    ADDL3   #1, FILENAME, DSTFAB+44         ; 7013
           00000000'  EF       04          BC   90 00010    MOVB    @FILENAME, DSTFAB+52            ; 7014
           00000000'  EF 00000000'         EF   9E 00018    MOVAB   DSTFAB, DSTRAB+60               ; 7015
           00000000'  EF                   6E   9E 00023    MOVAB   XBUFFER, DSTRAB+36              ; 7016
                      00000000'            EF   9F 0002A    PUSHAB  DSTFAB                          ; 7017
           00000000G  00                   01   FB 00030    CALLS   #1, SYS$OPEN                    ;
                      5A                   50   D0 00037    MOVL    R0, STATUS                      ;
                      09                   5A   E8 0003A    BLBS    STATUS, 1$                      ; 7018
                                           5A   DD 0003D    PUSHL   STATUS                          ;
           00000000G  00                   01   FB 0003F    CALLS   #1, LIB$SIGNAL                  ;
                           00000000'       EF   9F 00046 1$:  PUSHAB  DSTRAB                        ; 7019
           00000000G  00                   01   FB 0004C    CALLS   #1, SYS$CONNECT                 ;
                      5A                   50   D0 00053    MOVL    R0, STATUS                      ;
                      09                   5A   E8 00056    BLBS    STATUS, 2$                      ; 7020
                                           5A   DD 00059    PUSHL   STATUS                          ;
           00000000G  00                   01   FB 0005B    CALLS   #1, LIB$SIGNAL                  ;
                      5B                   01   CE 00062 2$:  MNEGL   #1, CURBLK                    ; 7027
                                           57   D4 00065    CLRL    DSTADDR                         ; 7028
                      59        08         AC   D0 00067    MOVL    IMG_DST_BLOCK, R9               ; 7029
                      50        08         A9   3C 0006B    MOVZWL  8(R9), R0                       ;
                50    50                   09   78 0006F    ASHL    #9, R0, R0                      ;
                      55        FF         A0   9E 00073    MOVAB   -1(R0), DSTENDADDR              ;
                54    57  00001400         8F   C7 00077 3$:  DIVL3   #5120, DSTADDR, BLKNUM          ; 7038
7E              00    57                   01   7A 0007F    EMUL    #1, DSTADDR, #0, -(SP)          ; 7039
58              58    8E  00001400         8F   7B 00084    EDIV    #5120, (SP)+, BYTNUM, BYTNUM    ;
                      5B                   54   D1 0008D    CMPL    BLKNUM, CURBLK                  ; 7040
                                           4E   13 00090    BEQL    6$                              ;
                50    54                   0A   C5 00092    MULL3   #10, BLKNUM, R0                 ; 7043
     00000000'  EF    50                   69   C1 00096    ADDL3   @(R9), R0, DSTRAB+56            ;
                50    54  00001400         8F   C5 0009E    MULL3   #5120, BLKNUM, R0               ; 7045
                50    55                   50   C3 000A6    SUBL3   R0, DSTENDADDR, R0              ;
```

```
                                                50  D6 000AA          INCL    R0                                          ;
                        00001600    8F          50  D1 000AC          CMPL    R0, #5632                                   ; 7044
                                                05  15 000B3          BLEQ    4$                                          ;
                                50      1600    8F  3C 000B5          MOVZWL  #5632, R0                                   ;
                        00000000'   EF          50  B0 000BA 4$:      MOVW    R0, DSTRAB+32                               ;
                                        00000000'   EF  9F 000C1      PUSHAB  DSTRAB                                      ; 7046
                        00000000G   00              01  FB 000C7      CALLS   #1, SYS$READ                                ;
                                    5A              50  D0 000CE      MOVL    R0, STATUS                                  ;
                                    09              5A  E8 000D1      BLBS    STATUS, 5$                                  ; 7047
                                                    5A  DD 000D4      PUSHL   STATUS                                      ;
                        00000000G   00              01  FB 000D6      CALLS   #1, LIB$SIGNAL                              ;
                                    5B              54  D0 000DD 5$:   MOVL    BLKNUM, CURBLK                              ; 7048
                    56              58              5E  C1 000E0 6$:   ADDL3   SP, BYTNUM, DSTPTR                         ; 7051
                                    55              57  D1 000E4      CMPL    DSTADDR, DSTENDADDR                         ; 7057
                                                    1E  1E 000E7      BGEQU   7$                                          ;
                                                    66  95 000E9      TSTB    (DSTPTR)                                    ; 7058
                                                    1A  13 000EB      BEQL    7$                                          ;
                                BC  8F      01      A6  91 000ED      CMPB    1(DSTPTR), #188                             ; 7059
                                                    17  12 000F2      BNEQ    8$                                          ;
                                    50      07      A6  9A 000F4      MOVZBL  7(DSTPTR), R0                               ; 7061
                                                    50  D6 000F8      INCL    R0                                          ;
            50          00 00000000'    EF          09  2D 000FA      CMPC5   #9, P.ABA, #0, R0, 7(DSTPTR)                ;
                                            07      A6      00103                                                       ;
                                                    04  12 00105      BNEQ    8$                                          ;
                                    50              57  D0 00107 7$:   MOVL    DSTADDR, R0                                 ; 7063
                                                    04  0010A      RET                                                 ;
                                    50              66  9A 0010B 8$:   MOVZBL  (DSTPTR), R0                               ; 7068
                                    57          01  A047  9E 0010E      MOVAB   1(R0)[DSTADDR], DSTADDR                     ;
                                                FF61  31 00113      BRW     3$                                          ; 7030

; Routine Size:  278 bytes,    Routine Base:  DBG$CODE + 31A0

; 6982          7077  1
; 6983          7078  0 END ELUDOM


                                                                    .EXTRN  LIB$SIGNAL, SYS$UNWIND

;                               PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;  DBG$GLOBAL                     56  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$OWN                     20356  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$PLIT                      527  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$CODE                    12982  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)


;                               Library Statistics
;
;                                       -------- Symbols --------      Pages       Processing
;       File                            Total    Loaded   Percent      Mapped      Time
```

```
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1              18619        85         0       1000        00:01.8
;  _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1         32         2         6          7        00:00.1
;  _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1         1545       144         9         97        00:02.1
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                               418       209        50         31        00:00.4
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1          386        30         7         22        00:00.3



; Information:  49
; Warnings:     0
; Errors:       0


;                                    COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RSTCNTRL/OBJ=OBJ$:RSTCNTRL MSRC$:RSTCNTRL/UPDATE=(ENH$:RSTCNTRL)

; Size:          12982 code + 20939 data bytes
; Run Time:         03:51.0
; Elapsed Time:     04:14.0
; Lines/CPU Min:     1838
; Lexemes/CPU-Min: 14035
; Memory Used:  962 pages
; Compilation Complete
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RSTCNTRL
LIS

0100 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RSTTYPES
LIS